1. Principais Bibliotecas do Google Apps Script

Essas bibliotecas já vêm embutidas no GAS. Você só precisa chamá-las diretamente no código.

Biblioteca	Função Principal	Caso de Uso no Seu Sistema
SpreadsheetApp	Manipular Google Sheets	CRUD completo em planilhas
DriveApp	Manipular Google Drive	Armazenar documentos, PDFs, imagens e logs
UrlFetchApp	Chamar APIs externas	OpenAI, Gemini, Bancos, Servidores Python
HtmlService	Criar UI personalizada	Painéis, dashboards, modais
ScriptApp	Triggers e automações internas	Agendamentos, workflows automáticos
PropertiesService	Guardar configurações seguras	API keys, tokens, credenciais
CacheService	Melhorar performance	Evitar chamadas repetitivas em APIs
MailApp / GmailApp	Enviar e-mails automáticos	Notificações de clientes ou equipe
CalendarApp	Integração com Google Calendar	Agendamentos de reuniões
FormsApp	Automação de formulários	Cadastrar clientes via formulários
Drive API Avançada	Recursos avançados do Drive	Controle detalhado de permissões

---

2. DriveApp – Armazenamento Interno

O **DriveApp** serve para **armazenar e gerenciar arquivos** do seu sistema.

Ideal para: **backup de dados**, **logs**, **relatórios em PDF**, **documentos de clientes**, **imagens**.

Exemplo: Criar uma pasta para cada cliente

```
function criarPastaCliente(nomeCliente) {
```

```
  const pastaRaiz = DriveApp.getFolderById("ID_DA_PASTA_PRINCIPAL");
  const novaPasta = pastaRaiz.createFolder(nomeCliente);
  return novaPasta.getUrl();
}
```

Exemplo: Salvar relatório em PDF dentro do Drive

```
function gerarRelatorioPDF() {
  const ss = SpreadsheetApp.getActiveSpreadsheet();
  const urlPDF = ss.getUrl().replace(/edit$/, '') + 'export?format=pdf';

  const options = {
    headers: { Authorization: 'Bearer ' + ScriptApp.getOAuthToken() }
  };

  const response = UrlFetchApp.fetch(urlPDF, options);
  const blob = response.getBlob().setName("Relatorio.pdf");

  DriveApp.getFolderById("ID_DA_PASTA_PRINCIPAL").createFile(blob);
}
```

Uso prático:

- Criar histórico mensal de contratos fechados em PDF.
- Gerar relatórios automáticos para diretoria.
- Salvar boletos e notas fiscais diretamente no Drive.

---

3. UrlFetchApp – Conectar com o Mundo Externo

Essa é **a ponte entre o seu sistema e APIs externas**, como OpenAI, Gemini, servidores Python, bancos ou ERPs externos.

Ideal para: **integração com IA**, **automação bancária**, **webhooks**, **serviços externos**.

Exemplo: Conectar com OpenAI

```
function chamarOpenAI(prompt) {
  const url = 'https://api.openai.com/v1/chat/completions';
  const apiKey = PropertiesService.getScriptProperties().getProperty("OPENAI_KEY");

  const payload = {
    model: 'gpt-4.1-mini',
    messages: [{ role: 'user', content: prompt }]
  };

  const options = {
    method: 'post',
    headers: {
      'Authorization': 'Bearer ' + apiKey,
      'Content-Type': 'application/json'
    },
    payload: JSON.stringify(payload),
    muteHttpExceptions: true
  };

  const response = UrlFetchApp.fetch(url, options);
  return JSON.parse(response.getContentText());
}
```

Uso prático:

- Gerar insights automáticos a partir de dados internos.
- Criar um chatbot interno conectado à sua planilha.
- Previsões avançadas com modelos Python hospedados externamente.

---

4. PropertiesService – Guardar Credenciais Seguras

Nunca deixe **API Keys** ou **tokens** diretamente no código.
Use o **PropertiesService** para segurança.

```
function salvarChaveOpenAI() {
  PropertiesService.getScriptProperties().setProperty("OPENAI_KEY", "SUA_CHAVE_AQUI");
}

function pegarChaveOpenAI() {
  return PropertiesService.getScriptProperties().getProperty("OPENAI_KEY");
}
```

Benefícios:

- Segurança: evita expor chaves diretamente no script.
- Facilidade: se precisar trocar a chave, só altera no painel de propriedades.

---

5. CacheService – Melhorar Performance

Se você faz muitas chamadas a APIs externas ou consultas repetitivas, pode usar **CacheService** para guardar resultados temporários.

Ideal para: dashboards, relatórios, integrações com IA.

Exemplo: Guardar dados por 5 minutos

```
function consultarDados() {
  const cache = CacheService.getScriptCache();
  const dadosCache = cache.get("dados_api");

  if (dadosCache) {
    return JSON.parse(dadosCache);
  }
```

```
  const dadosNovos =
UrlFetchApp.fetch("https://api.exemplo.com/dados").getContentText();
  cache.put("dados_api", dadosNovos, 300); // 300 segundos = 5 minutos
  return JSON.parse(dadosNovos);
}
```

Uso prático:

- Evitar bater na API OpenAI/Gemini várias vezes.
- Melhorar velocidade de dashboards.
- Reduzir custos com requisições externas.

---

6. ScriptApp – Automação Interna com Triggers

Automatize processos sem depender de ferramentas externas como Zapier ou n8n.

Trigger	Quando dispara	Caso de uso
onEdit	Quando algo muda na planilha	Atualizar status de contrato
onFormSubmit	Quando um Google Form é enviado	Cadastrar novo cliente
Time-driven	Em horários definidos	Relatórios diários/semanais
onOpen	Quando o Google Sheets abre	Configurações iniciais

Exemplo: Enviar relatório diário automaticamente

```
function enviarRelatorioDiario() {
  const relatorio = gerarRelatorioPDF();
  MailApp.sendEmail({
    to: "diretoria@empresa.com",
    subject: "Relatório Diário",
    body: "Segue o relatório em anexo",
    attachments: [relatorio]
  });
}
```

Depois vá em **Editar > Acionadores > Novo acionador**
e configure para rodar **todo dia às 8h**.

---

7. Combinação: Sheets + Drive + IA

Aqui está um fluxo poderoso:

1. Dados coletados no **Google Sheets** (planilhas internas).
2. Relatório gerado e salvo no **Google Drive** via DriveApp.
3. Dados processados e analisados com **OpenAI ou Gemini** via UrlFetchApp.
4. Relatório final enviado automaticamente para a diretoria via **MailApp**.

Exemplo final

```
function fluxoCompleto() {
  const dados = SpreadsheetApp.getActiveSpreadsheet()
    .getSheetByName("Contratos")
    .getDataRange()
    .getValues();

  const resumo = chamarOpenAI("Analise esses contratos e gere insights: " +
JSON.stringify(dados));

  const blob = Utilities.newBlob(resumo.choices[0].message.content, "text/plain",
"resumo.txt");
  DriveApp.getFolderById("ID_DA_PASTA_PRINCIPAL").createFile(blob);

  MailApp.sendEmail({
    to: "diretoria@empresa.com",
    subject: "Relatório Inteligente",
    body: "Segue em anexo o relatório com insights gerados por IA.",
    attachments: [blob]
  });
```

}

---

8. Estrutura Final Integrada

[Usuário]

|

v

[Front-end: HTML + CSS + JS]

|

v

[Google Apps Script]

|

+--> [SpreadsheetApp] → Dados internos

+--> [DriveApp] → Relatórios, PDFs, arquivos

+--> [UrlFetchApp] → APIs externas (OpenAI, Gemini, Python)

+--> [CacheService] → Performance

+--> [PropertiesService] → Segurança

+--> [ScriptApp] → Automação

|

v

[Relatórios + IA + Automação completa]