

♦ 1. Importações

```
from flask import Flask, render_template, request, redirect,
url_for, session
from flask_login import LoginManager, login_user, logout_user,
login_required, UserMixin, current_user

import json
import os
```

✅ O que faz:

- `Flask`, `render_template`, etc.: usados para lidar com páginas, formulários, redirecionamentos, sessões.
- `flask_login`: biblioteca que gerencia login/logout de usuários.
- `json` e `os`: usados para **ler e gravar** os arquivos `.json` com produtos, carrinhos e compras.

⚠ **Erro comum:** se não instalar o `flask-login`, dará erro. Use:

```
pip install flask flask-login
```

♦ 2. Configuração do Flask e login

```
app = Flask(__name__)
app.config['SECRET_KEY'] = 'RANCA TAMPA E MANDA BOI'
```

✅ O que faz:

- Cria o app Flask.
 - Define uma **chave secreta** para proteger sessões e cookies.
-

♦ 3. Login Manager do Flask-Login

```
login_manager = LoginManager(app)
login_manager.login_view = 'login'
```

✓ O que faz:

- Cria um gerenciador de login.
- Diz que, se o usuário não estiver logado, deve ser mandado para a rota 'login'.

◆ 4. Variáveis de arquivos e dados

```
usuarios = {}
```

```
PRODUTOS_ARQ = 'produtos.json'
CARRINHOS_ARQ = 'carrinhos.json'
COMPRAS_ARQ = 'compras.json'
```

✓ O que faz:

- Guarda os nomes dos arquivos JSON usados.
- `usuarios` guarda nomes e senhas, **somente na memória**.

⚠ **Melhoria:** os usuários somem quando você reinicia o app. Se quiser persistir, salve isso em um arquivo ou banco.

◆ 5. Carregar produtos e carrinhos

```
if os.path.exists(PRODUTOS_ARQ):
    with open(PRODUTOS_ARQ, 'r') as f:
        produtos_ = json.load(f)
else:
    produtos_ = {}

if os.path.exists(CARRINHOS_ARQ):
    with open(CARRINHOS_ARQ, 'r') as f:
```

```
        carrinhos = json.load(f)
else:
    carrinhos = {}
```

✓ O que faz:

- Lê os produtos e carrinhos salvos nos arquivos `.json`.
 - Se o arquivo não existir, cria um dicionário vazio.
-

♦ 6. Função para salvar carrinhos

```
def salvar_carrinhos():
    with open(CARRINHOS_ARQ, 'w') as f:
        json.dump(carrinhos, f, indent=4)
```

✓ O que faz:

- Atualiza o arquivo `carrinhos.json` com o que tem no dicionário `carrinhos`.
-

♦ 7. Classe de Usuário

```
class User(UserMixin):
    def __init__(self, username):
        self.id = username
```

✓ O que faz:

- Define um usuário compatível com `flask_login`.
 - O `.id` é o que o Flask-Login usa para identificar o usuário.
-

♦ 8. Função `load_user`

```
@login_manager.user_loader
def load_user(user_id):
    if user_id in usuarios:
        return User(user_id)
    return None
```

✓ O que faz:

- O Flask chama essa função automaticamente para recarregar o usuário logado.
-

♦ 9. Rota / (página inicial)

```
@app.route('/')
def index():
    return render_template('index.html')
```

✓ Mostra a página inicial.

♦ 10. Login de usuários

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        nome = request.form['nome']
        senha = request.form['senha']
        if nome in usuarios and usuarios[nome] == senha:
            user = User(nome)
            login_user(user)
            carrinhos.setdefault(nome, [])
            return redirect(url_for('produtos'))
        return redirect(url_for('login'))
    return render_template('login.html')
```

✓ Explicação:

- Se for **POST**, ele tenta logar o usuário.

- Se o nome e senha baterem, ele faz login e garante que tenha um carrinho.
- Redireciona para `/produtos`.

⚠ **Melhoria:** o login não dá mensagem de erro, só recarrega a página.

♦ 11. Cadastro

```
@app.route('/cadastro', methods=['GET', 'POST'])
def cadastro():
    if request.method == 'POST':
        nome = request.form['nome']
        senha = request.form['senha']
        if nome not in usuarios:
            usuarios[nome] = senha
        return redirect(url_for('login'))
    return render_template('cadastro.html')
```

✅ **Cria usuário novo.**

⚠ **Problema:** se você cadastrar alguém e reiniciar o app, ele **some**, porque não salva em arquivo.

♦ 12. Logout

```
@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))
```

✅ **Desloga o usuário atual.**

♦ 13. Página de produtos

```
@app.route('/produtos')
```

```
@login_required
def produtos():
    return render_template('produtos.html', produtos=produtos_)
```

✓ Mostra os produtos para o usuário logado.

♦ 14. Adicionar ao carrinho

```
@app.route('/adicionar', methods=['POST'])
@login_required
def adicionar():
    prod = request.form['prod']
    carrinhos.setdefault(current_user.id, []).append(prod)
    salvar_carrinhos()
    return redirect(url_for('carrinho'))
```

✓ Adiciona o produto no carrinho do usuário atual.

⚠ **Erro comum:** se o nome do produto não estiver em `produtos_`, o valor no carrinho não será somado corretamente.

♦ 15. Remover do carrinho

```
@app.route('/remover', methods=['POST'])
@login_required
def remover():
    prod = request.form['prod']
    carrinho = carrinhos.get(current_user.id, [])
    if prod in carrinho:
        carrinho.remove(prod)
        salvar_carrinhos()
    return redirect(url_for('carrinho'))
```

✓ Remove um item do carrinho.

♦ 16. Página do carrinho

```
@app.route('/carrinho')
@login_required
def carrinho():
    carrinho_ = carrinhos.get(current_user.id, [])
    soma = sum(produtos_[p] for p in carrinho_)
    return render_template('carrinho.html', carrinho=carrinho_,
valor=soma, produtos=produtos_)
```

✅ **Mostra o carrinho do usuário, com o valor total.**

⚠ **Erro comum:** se um produto foi removido do `produtos.json` mas ainda está no carrinho, pode dar erro de `KeyError`.

♦ 17. Fechar o carrinho (compra finalizada)

```
@app.route('/fechar_carrinho', methods=['POST'])
@login_required
def fechar_carrinho():
    usuario = current_user.id
    carrinho_ = carrinhos.get(usuario, [])

    if not carrinho_:
        return redirect(url_for('carrinho'))

    if os.path.exists(COMPRAS_ARQ):
        with open(COMPRAS_ARQ, 'r') as f:
            historico = json.load(f)
    else:
        historico = {}

    historico.setdefault(usuario, [])
    historico[usuario].append(carrinho_)

    with open(COMPRAS_ARQ, 'w') as f:
        json.dump(historico, f, indent=4)

    carrinhos[usuario] = []
    salvar_carrinhos()
```

```
return redirect(url_for('produtos'))
```

✅ Finaliza a compra:

- Lê o histórico,
- Salva o carrinho atual nele,
- Limpa o carrinho,
- Salva tudo de volta.



1. Instalar e usar a biblioteca **werkzeug.security**

Ela já vem com o Flask e nos dá duas funções essenciais:

```
from werkzeug.security import generate_password_hash,  
check_password_hash
```

Função	O que faz
<code>generate_password_hash(senha)</code>	Cria uma versão criptografada da senha
<code>check_password_hash(hash, senha_digitada)</code>	Compara se a senha digitada bate com o hash salvo



2. Salvar usuários em um arquivo **usuarios.json**

Estrutura esperada do arquivo:

```
{  
  "isabele": "HASH_DA_SENHA",  
  "tata": "OUTRO_HASH"  
}
```



3. Atualize seu código passo a passo

Passo 1: Crie o nome do arquivo de usuários

```
python
CopiarEditar
USUARIOS_ARQ = 'usuarios.json'
```

Passo 2: Carregue os usuários ao iniciar

```
python
CopiarEditar
if os.path.exists(USUARIOS_ARQ):
    with open(USUARIOS_ARQ, 'r') as f:
        usuarios = json.load(f)
else:
    usuarios = {}
```

Passo 3: Crie função para salvar os usuários

```
python
CopiarEditar
def salvar_usuarios():
    with open(USUARIOS_ARQ, 'w') as f:
        json.dump(usuarios, f, indent=4)
```



4. Alterar o cadastro para salvar senha com segurança

Antigo:

```
python
CopiarEditar
if nome not in usuarios:
    usuarios[nome] = senha
```

Novo (com hash + salvar):

```
python
CopiarEditar
if nome not in usuarios:
    hash_senha = generate_password_hash(senha)
    usuarios[nome] = hash_senha
    salvar_usuarios()
```

5. Alterar o login para verificar hash

Antigo:

python

CopiarEditar

```
if nome in usuarios and usuarios[nome] == senha:
```

Novo:

python

CopiarEditar

```
if nome in usuarios and check_password_hash(usuarios[nome], senha):
```

Resultado:

- Os usuários são **salvos no arquivo `usuarios.json`**.
- As senhas **nunca ficam visíveis**, só os *hashes criptografados*.
- O sistema fica **muito mais seguro**.

1. Importações que você precisa no topo

Adicione isso no topo do seu `app.py`:

python

CopiarEditar

```
from werkzeug.security import generate_password_hash,  
check_password_hash  
from flask import flash
```

2. Variável e função para usuários

Logo depois da parte que define `PRODUTOS_ARQ`, `CARRINHOS_ARQ` e `COMPRAS_ARQ`, adicione:

```
python
CopiarEditar
USUARIOS_ARQ = 'usuarios.json'

# Carrega os usuários do arquivo
if os.path.exists(USUARIOS_ARQ):
    with open(USUARIOS_ARQ, 'r') as f:
        usuarios = json.load(f)
else:
    usuarios = {}

# Salvar os usuários no arquivo
def salvar_usuarios():
    with open(USUARIOS_ARQ, 'w') as f:
        json.dump(usuarios, f, indent=4)
```

✓ 3. Corrigir o **/cadastro** com criptografia e **flash**

Substitua **todo o bloco /cadastro** por este:

```
python
CopiarEditar
@app.route('/cadastro', methods=['GET', 'POST'])
def cadastro():
    if request.method == 'POST':
        nome = request.form['nome'].strip()
        senha = request.form['senha']

        if nome in usuarios:
            flash('Esse nome de usuário já está cadastrado.',
'erro')
            return redirect(url_for('cadastro'))

        hash_senha = generate_password_hash(senha)
        usuarios[nome] = hash_senha
        salvar_usuarios()
        flash('Cadastro realizado com sucesso! Faça login.',
'sucesso')
        return redirect(url_for('login'))
```

```
return render_template('cadastro.html')
```

✓ 4. Corrigir o `/login` para verificar o hash e exibir erros

Substitua **todo o bloco `/login`** por este:

python

CopiarEditar

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        nome = request.form['nome'].strip()
        senha = request.form['senha']

        if nome in usuarios and check_password_hash(usuarios[nome],
senha):
            user = User(nome)
            login_user(user)
            carrinhos.setdefault(nome, [])
            flash(f'Bem-vindo(a), {nome}!', 'sucesso')
            return redirect(url_for('produtos'))

        flash('Usuário ou senha incorretos.', 'erro')
        return redirect(url_for('login'))

    return render_template('login.html')
```

✓ 5. Exibir os `flash()` nas páginas HTML

No seu `base.html`, ou no topo de cada template (tipo `login.html`, `cadastro.html`), adicione:

html

CopiarEditar

```
{% with mensagens = get_flashed_messages(with_categories=True) %}
{% if mensagens %}
<ul class="mensagens">
```

```
{% for categoria, msg in mensagens %}
  <li class="{{ categoria }}">{{ msg }}</li>
{% endfor %}
</ul>
{% endif %}
{% endwith %}
```

E no seu CSS, pode fazer algo assim:

css

CopiarEditar

```
.mensagens { list-style: none; padding: 0; }
.mensagens .erro { color: red; }
.mensagens .sucesso { color: green; }
```

✓ Pronto! Agora o sistema:

- Salva os usuários com senha **criptografada**.
- Guarda tudo no arquivo **usuarios.json**.
- Dá **mensagens de erro e sucesso** com **flash()**.
- E protege contra nomes repetidos no cadastro!

✓ Como usar no HTML (**base.html** ou no topo de **login.html**, etc.)

html

CopiarEditar

```
{% with mensagens = get_flashed_messages(with_categories=True) %}
{% if mensagens %}
  <ul class="mensagens">
    {% for categoria, msg in mensagens %}
      <li class="{{ categoria }}">{{ msg }}</li>
    {% endfor %}
  </ul>
{% endif %}
{% endwith %}
.mensagens {
```

```
list-style: none;
padding: 0;
margin: 10px 0;
}

.mensagens .sucesso {
color: green;
background-color: #e6ffe6;
border: 1px solid green;
padding: 10px;
border-radius: 5px;
}

.mensagens .erro {
color: red;
background-color: #ffe6e6;
border: 1px solid red;
padding: 10px;
border-radius: 5px;
}
```

✓ 1. **session** (sessões) — JÁ ESTÁ USANDO indiretamente

✓ Já usado no seu código:

```
python
CopiarEditar
from flask import session
```

O **Flask-Login** já usa sessões por baixo dos panos para manter o usuário logado.

✓ Como usar **session** manualmente (pode cair na prova!):

```
python
CopiarEditar
session['nome'] = 'isabele'
```

```
python
CopiarEditar
nome = session.get('nome')
```

```
python
```

CopiarEditar

```
session.pop('nome', None) # remove da sessão
```

📌 **Exemplo prático:** salvar o nome do último produto que o usuário clicou:

python

CopiarEditar

```
@app.route('/detalhe/<produto>')
def detalhe(produto):
    session['ultimo_produto'] = produto
    return f"Você viu o produto {produto}"
```

✅ 2. flash() — VOCÊ JÁ INSERIU

Você já está usando corretamente com:

python

CopiarEditar

```
flash('Mensagem', category='erro')
```

E exibindo no HTML com:

html

CopiarEditar

```
{% with mensagens = get_flashed_messages(with_categories=True) %}
```

✅ 3. Cookies — AINDA NÃO ESTÁ USANDO diretamente

Cookies são diferentes de sessão: eles **ficam no navegador**.

Como salvar um cookie:

python

CopiarEditar

```
from flask import make_response

@app.route('/lembrar_usuario/<nome>')
def lembrar_usuario(nome):
```

```
resposta = make_response(redirect(url_for('index')))
resposta.set_cookie('usuario_salvo', nome)
return resposta
```

Como ler um cookie:

```
python
CopiarEditar
@app.route('/')
def index():
    nome = request.cookies.get('usuario_salvo')
    return render_template('index.html', nome=nome)
```

✅ 4. String de consulta (query strings) — NÃO ESTÁ USANDO

📌 **O que é?:** São os parâmetros que vêm após `?` na URL.

Exemplo de URL:

```
/produtos?categoria=bota&preco_max=1000
```

Como capturar:

```
python
CopiarEditar
@app.route('/produtos')
def produtos():
    categoria = request.args.get('categoria')
    preco_max = request.args.get('preco_max')
    return f"Filtrando por: {categoria}, até R$ {preco_max}"
```

✅ 5. Segurança extra (proteção contra erros comuns)

Sua app está segura nas rotas com `@login_required` ✅

Mas você pode adicionar:

```
python
CopiarEditar
@app.errorhandler(404)
def pagina_nao_encontrada(e):
```



```
return render_template('404.html'), 404
```

✓ RESUMO DO QUE ESTÁ E DO QUE PODE CAIR

Assunto	Está no seu código?	Pode cair?	Exemplo/Teste
<code>session</code> (login, dados)	✓	✓	<code>session['chave'] = valor</code>
<code>flash()</code>	✓	✓	<code>flash('msg', 'erro')</code> + exibição com Jinja
Cookies (<code>set_cookie</code>)	✗	✓	<code>resposta.set_cookie(...)</code>
Query strings (<code>?x=1</code>)	✗	✓	<code>request.args.get(...)</code>
Criptografia com <code>hash</code>	✓ (após ajuste)	✓	<code>generate_password_hash</code> , <code>check_password_hash</code>
<code>@login_required</code>	✓	✓	Protege rotas como <code>/produtos</code>



Quer praticar?

Posso montar uma mini-provinha de múltipla escolha, verdadeiro/falso e uma dissertativa, pra você treinar.

Quer que eu faça isso agora rapidinho?

Você disse:

A prova vai ser fazendo código. Me mostre como eu usaria tudo isso aí em uma possível prova por favor. De forma simples, e que pareça que foi um aluno que fez. E explique passo a passo

O ChatGPT disse:

Perfeito, Isabele! Vamos montar um **exemplo de código simples e completo**, como se você estivesse respondendo uma **questão prática de prova**, usando:

- `session`

- `flash()`
 - `cookies`
 - `query strings`
 - `Flask-Login`
 - `senha criptografada`
-

ENUNCIADO DE PROVA (exemplo que poderia cair):

Crie uma aplicação Flask com:

- Página de **cadastro** com senha criptografada.
 - Página de **login** com `flash` de erro se usuário ou senha estiverem incorretos.
 - Após login, mostre uma página `/perfil` com o nome do usuário.
 - Salve o nome em `session`.
 - Crie um botão que ativa a função "lembrar meu nome" salvando em **cookie**.
 - Mostre no `/` uma mensagem com o nome salvo no cookie (se houver).
 - Use uma **query string** para mostrar uma mensagem de boas-vindas com `?mensagem=...`
-

RESPOSTA DE UMA ALUNA (COMENTADO E SIMPLES):

python

CopiarEditar

```
from flask import Flask, render_template, request, redirect, url_for, session, flash, make_response
```

```

from werkzeug.security import generate_password_hash,
check_password_hash
from flask_login import LoginManager, login_user, logout_user,
login_required, UserMixin, current_user
import json
import os

app = Flask(__name__)
app.secret_key = 'minha-chave-secreta'

login_manager = LoginManager(app)
login_manager.login_view = 'login'

# Usuários salvos em arquivo
USUARIOS_ARQ = 'usuarios.json'

if os.path.exists(USUARIOS_ARQ):
    with open(USUARIOS_ARQ, 'r') as f:
        usuarios = json.load(f)
else:
    usuarios = {}

def salvar_usuarios():
    with open(USUARIOS_ARQ, 'w') as f:
        json.dump(usuarios, f)

# Classe de usuário para Flask-Login
class User(UserMixin):
    def __init__(self, nome):
        self.id = nome

@login_manager.user_loader
def load_user(user_id):
    if user_id in usuarios:
        return User(user_id)

# Página inicial com query string e cookie
@app.route('/')
def index():
    mensagem = request.args.get('mensagem') # Ex: /?mensagem=Olá
    nome_cookie = request.cookies.get('lembrado')

```

```
        return f"<h2>{mensagem or ''}</h2><p>Nome lembrado: {nome_cookie  
or 'Nenhum nome salvo'}</p>"
```

```
# Cadastro de usuário com senha criptografada
```

```
@app.route('/cadastro', methods=['GET', 'POST'])
```

```
def cadastro():
```

```
    if request.method == 'POST':
```

```
        nome = request.form['nome']
```

```
        senha = request.form['senha']
```

```
        if nome in usuarios:
```

```
            flash('Usuário já existe!', 'erro')
```

```
            return redirect(url_for('cadastro'))
```

```
        usuarios[nome] = generate_password_hash(senha)
```

```
        salvar_usuarios()
```

```
        flash('Cadastro feito! Faça login.', 'sucesso')
```

```
        return redirect(url_for('login'))
```

```
    return render_template('cadastro.html')
```

```
# Login com flash de erro
```

```
@app.route('/login', methods=['GET', 'POST'])
```

```
def login():
```

```
    if request.method == 'POST':
```

```
        nome = request.form['nome']
```

```
        senha = request.form['senha']
```

```
        if nome in usuarios and check_password_hash(usuarios[nome],  
senha):
```

```
            user = User(nome)
```

```
            login_user(user)
```

```
            session['nome'] = nome # salva nome na sessão
```

```
            flash('Login feito com sucesso!', 'sucesso')
```

```
            return redirect(url_for('perfil'))
```

```
        flash('Usuário ou senha incorretos!', 'erro')
```

```
        return redirect(url_for('login'))
```

```
    return render_template('login.html')
```

```
# Página do perfil (protegida)
```

```
@app.route('/perfil')
```

```
@login_required
```

```
def perfil():
```

```
    nome = session.get('nome')
```

```

        return f"<h1>Bem-vindo(a), {nome}!</h1><a
href='/lembrar'>Lembrar meu nome</a>"

# Salva nome em cookie
@app.route('/lembrar')
@login_required
def lembrar():
    nome = current_user.id
    resp = make_response(redirect(url_for('index')))
    resp.set_cookie('lembrado', nome)
    return resp

# Logout
@app.route('/logout')
@login_required
def logout():
    logout_user()
    session.clear()
    flash('Você saiu da conta.', 'sucesso')
    return redirect(url_for('login'))

```



Templates simples (como se fosse de prova)

♦ login.html

html

CopiarEditar

```
<h1>Login</h1>
```

```

{% with mensagens = get_flashed_messages(with_categories=True) %}
    {% for categoria, msg in mensagens %}
        <p class="{{ categoria }}">{{ msg }}</p>
    {% endfor %}
{% endwith %}

```

```

<form method="post">
    <input name="nome" placeholder="Nome"><br>
    <input type="password" name="senha" placeholder="Senha"><br>
    <button type="submit">Entrar</button>
</form>

```

♦ cadastro.html

html

CopiarEditar

```
<h1>Cadastro</h1>
```

```
{% with mensagens = get_flashed_messages(with_categories=True) %}
  {% for categoria, msg in mensagens %}
    <p class="{{ categoria }}">{{ msg }}</p>
  {% endfor %}
{% endwith %}
```

```
<form method="post">
  <input name="nome" placeholder="Nome"><br>
  <input type="password" name="senha" placeholder="Senha"><br>
  <button type="submit">Cadastrar</button>
</form>
```

✓ O que esse código cobre:

Tema	Usado aqui?	Linha/explicação
session	✓	session['nome'] = nome
flash()	✓	flash('mensagem', 'categoria')
cookies (set_cookie)	✓	resp.set_cookie('lembrado', nome)
query string (?msg=oi)	✓	request.args.get('mensagem',)
check_password_hash	✓	Senhas seguras
Flask-Login	✓	login_user, @login_required



EXEMPLOS PRONTOS COM APLICAÇÕES DIFERENTES



1. Loja com carrinho de compras

Assuntos que caem:

- `login_required` para comprar
- Carrinho guardado com `session` ou em JSON
- `flash` para avisar que produto foi adicionado
- Botão para "fechar compra" (salvar em arquivo ou banco)

Código rápido de adicionar ao carrinho:

python

CopiarEditar

```
@app.route('/adicionar/<produto>')
@login_required
def adicionar(produto):
    carrinho = session.get('carrinho', [])
    carrinho.append(produto)
    session['carrinho'] = carrinho
    flash(f'{produto} adicionado ao carrinho!', 'sucesso')
    return redirect(url_for('produtos'))
```



2. Sistema de Hospital (médicos, pacientes, prontuário)

Assuntos que caem:

- Login de médico ou atendente
- Sessão guarda quem está logado
- Cookies podem lembrar o paciente que foi atendido por último
- Query string pra buscar por CPF, por exemplo: `/paciente?cpf=123`

python

CopiarEditar

```
@app.route('/paciente')
@login_required
def paciente():
    cpf = request.args.get('cpf')
    return f"Buscando prontuário do CPF: {cpf}"
```

3. Biblioteca (cadastro de usuários e livros)

Assuntos que caem:

- `flash()` para dizer que o livro foi emprestado
- `session` guarda o usuário que está com livros
- `@login_required` em tudo
- Cookie para lembrar o último livro emprestado

```
python
CopiarEditar
@app.route('/emprestar/<livro>')
@login_required
def emprestar(livro):
    session.setdefault('livros', []).append(livro)
    flash(f'Você pegou o livro "{livro}"', 'sucesso')
    resp = make_response(redirect(url_for('perfil')))
    resp.set_cookie('ultimo_livro', livro)
    return resp
```

Como adaptar qualquer tema na hora da prova:

Se cair...	Pense em...
Loja	Produtos, carrinho, compra final
Hospital	Médicos, pacientes, prontuário, agendamento
Biblioteca	Livros, empréstimo, devolução, usuários
Escola	Professores, alunos, notas, histórico

✓ DICA DE OURO: USE ESTA BASE SEMPRE

1. Cadastro com senha criptografada ✓
2. Login com `flash()` de erro ✓
3. Sessão com dados do usuário ✓
4. `@login_required` nas rotas privadas ✓
5. `flash()` para avisar ações ✓
6. `cookies` para lembrar algo ✓
7. `query strings` para filtros ou buscas ✓

Se quiser, posso montar um modelo prático baseado em qual desses temas você quiser:

- Loja de vaquejada?
- Prontuário de hospital?
- Biblioteca escolar?

Só dizer: “**quero o modelo de biblioteca!**” que eu monto tudinho pra você testar e estudar.