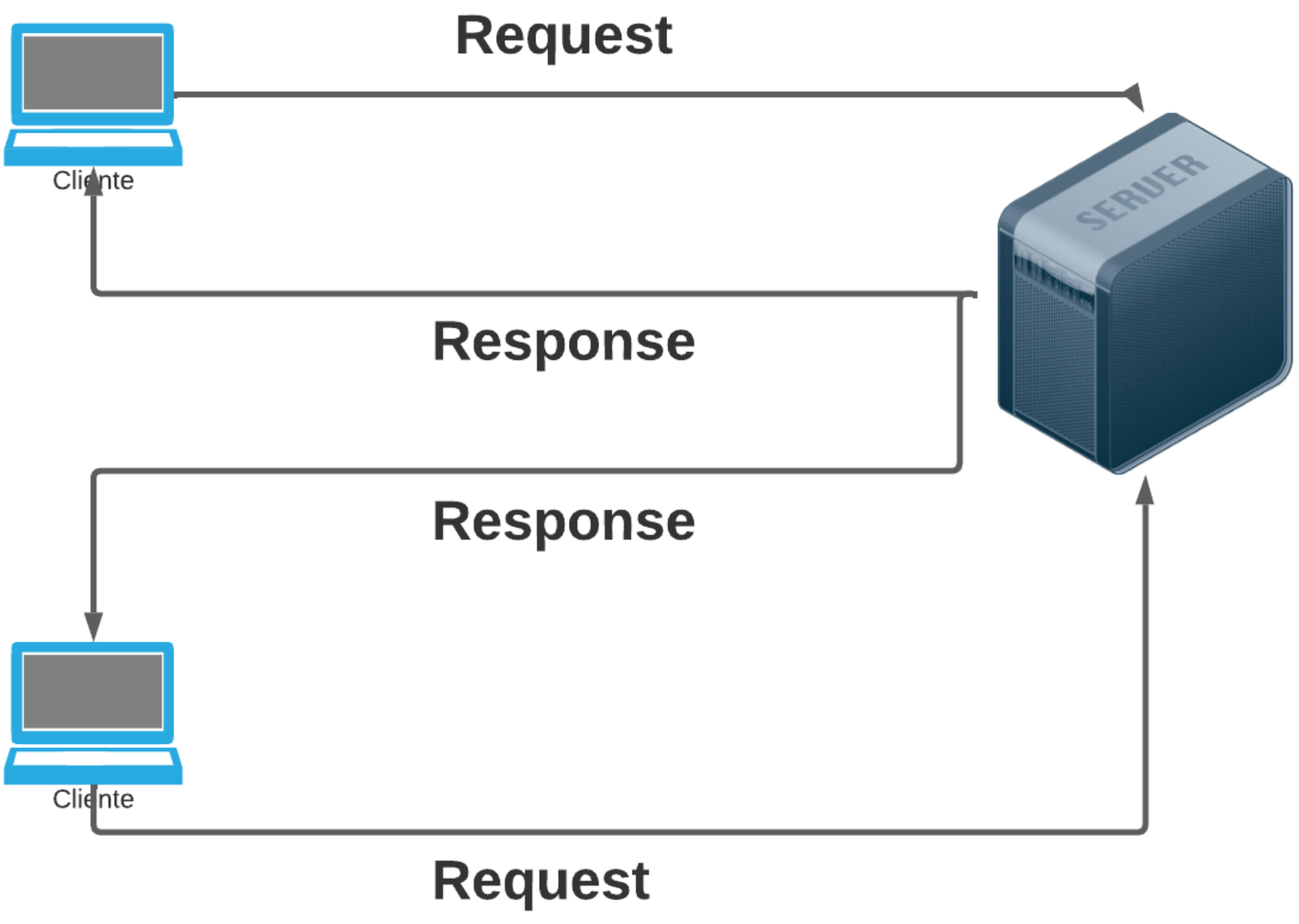
PSW 8.0 | Aula 1

Acesse pelo notion:

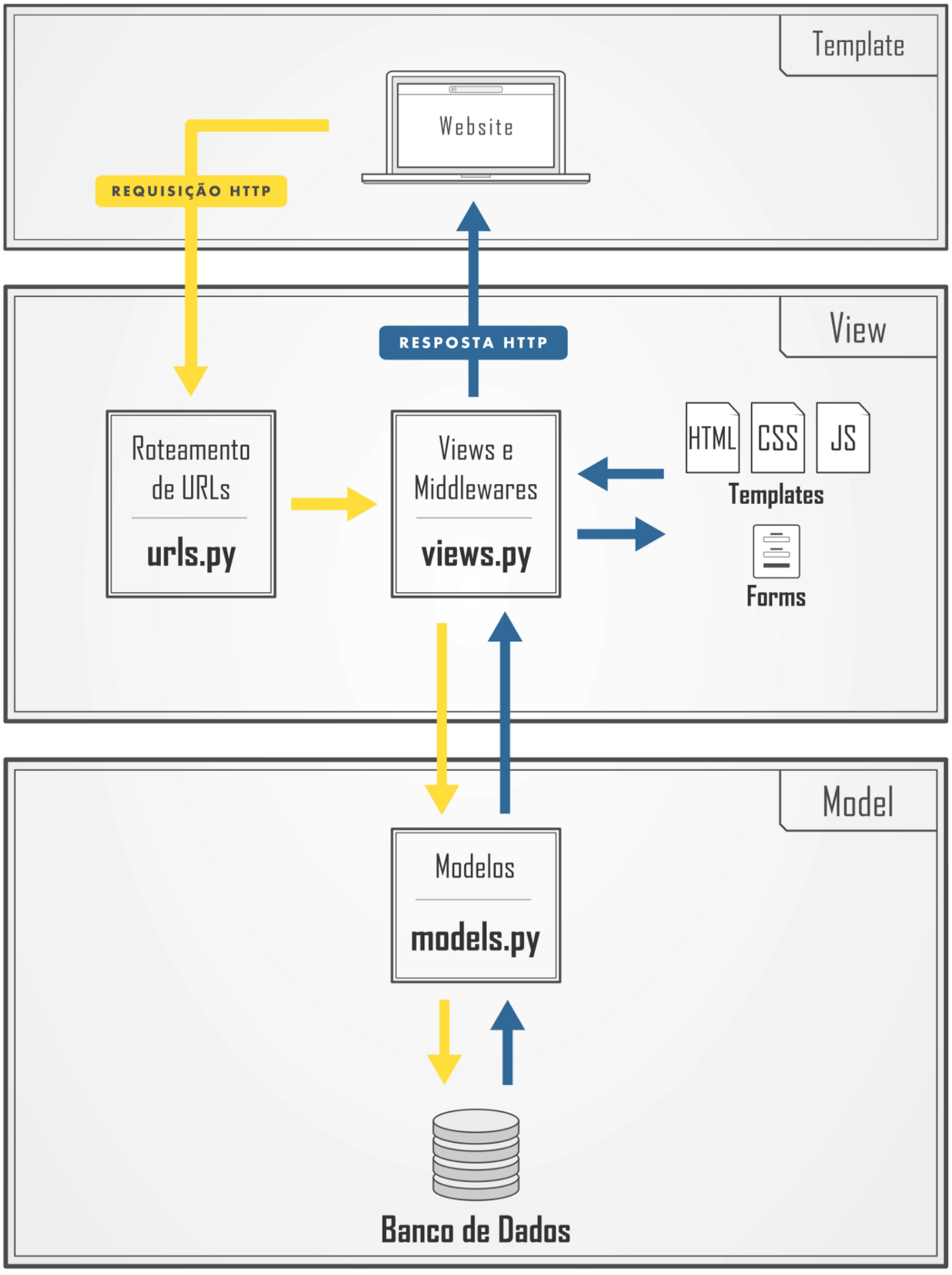
```
https://grizzly-amaranthus-f6a.notion.site/PSW-8-0-Aula-1-c5f28c09c09f4493ad20911f984e4fc8?pvs=4
```

▼ Conceitos



Fluxo de dados no Django:

ARQUITETURA DO django



▼ O projeto

https://www.figma.com/file/FzqXqJXe5a8LWcq7LxISHN/Untitled?type=design&node-id=0%3A1&mode=design&t=yK6MM8FJhwLIGMhE-1

▼ Configurações iniciais

Primeiro devemos criar o ambiente virtual:

```
# Criar
  # Linux
    python3 -m venv venv
  # Windows
    python -m venv venv
```

Após a criação do venv vamos ativa-lo:

```
#Ativar
  # Linux
    source venv/bin/activate
  # Windows
    venv\Scripts\Activate

# Caso algum comando retorne um erro de permissão execute o código e tente novamente:

Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

Agora vamos fazer a instalação do Django e as demais bibliotecas:

```
pip install django
pip install pillow
```

Vamos criar o nosso projeto Django:

```
django-admin startproject vitalab .
```

Rode o servidor para testar:

```
python manage.py runserver
```

Crie o app usuario:

```
python manage.py startapp usuarios
```

INSTALE O APP!

▼ Cadastro

Crie a URL para usuarios:

```
path('usuarios/', include('usuarios.urls')),
```

Agora crie o arquivo urls.py dentro de usuarios:

```
from django.urls import path
from . import views

urlpatterns = [
    path('cadastro/', views.cadastro, name="cadastro"),
]
```

Crie a função cadastro em views.py:

```
def cadastro(request):
    if request.method == "GET":
        return render(request, 'cadastro.html')
```

Configure onde o Django irá procurar por arquivos .html:

```
os.path.join(BASE_DIR, 'templates')
```

Crie o templates/bases/base.html:

```
{% load static %}
<!doctype html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>VitaLab</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet">

    {% block 'head' %}{% endblock %}
  </head>
  <body>

    {% block 'conteudo' %}{% endblock %}
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js"></script>
  </body>
</html>
```

Agora vamos criar o cadastro.html

```
{% extends "bases/base.html" %}
{% load static %}

{% block 'head' %}



{% endblock 'head' %}



{% block 'conteudo' %}
```

```
    <br>
    <br>
    <div class="container">
        <h3 class="font-destaque">Cadastre-se</h3>

        <div class="row">
            <div class="col-md-3" style="text-align: center">
                <img src="" alt="">
                <h3>VitaLab</h3>
            </div>

            <div class="col-md-9">
                <form action="" method="POST">
                <label>Primeiro nome</label>
                <br>
                <input type="text" class="input-default" name="primeiro_nome">
                <br>
                <br>
                <label>Último nome</label>
                <br>
                <input type="text" class="input-default" name="ultimo_nome">
            </div>
        </div>

        <div class="row">
            <div class="col-md-4">
                <label>Username</label>
                <br>
                <input type="text" class="input-default w100" name="username">
                <br>
                <br>
                <label>Senha</label>
                <br>
                <input type="text" class="input-default w100" name="senha">
            </div>

            <div class="col-md-4">
                <label>E-mail</label>
                <br>
                <input type="text" class="input-default w100" name="email">
                <br>
                <br>
                <label>Confirmar senha</label>
                <br>
                <input type="text" class="input-default w100" name="confirmar_senha">
            </div>

        </div>
        <br>
        <input type="submit" class="btn-default">
    </form>
    </div>
{% endblock %}
```

Nessa etapa, precisamos estilizar nossa aplicação criando os css. Para isso vamos configurar os arquivos estáticos:

```
STATIC_URL = '/static/'
STATICFILES_DIRS = (os.path.join(BASE_DIR, 'templates/static'),)
STATIC_ROOT = os.path.join('static')

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
```

em templates/static/geral/css/base.css crie o arquivo:

```
:root{

    --main-color: #151C34;
    --dark-color: #0C121C ;
    --light-color: #6DE6EE;
    --contrast-color: #f4c96b;
    --differential-color: #066668;

}
*{
```

```
    color: white
}

.font-destaque{
    color: var(--light-color);
    font-size: 40px;
}

.input-default{
    background-color: rgba(255,255,255,0.05);
    border: 1px solid var(--differential-color);
    padding: 7px;
    width: 50%;

}

.w100{
    width: 100%;
}

.btn-default{
    background-color: var(--light-color);
    color: black;
    width: 15%;
    padding: 10px;
    border: none;
    border-radius: 10px;
}

.font-destaque-secundaria{
    color: var(--light-color);
    font-size: 35px;
}
```

Agora é só importar o arquivo em base.html:

```
<link href="{% static 'geral/css/base.css' %}" rel="stylesheet">
```

Crie o arquivo templates/static/usuarios/css/css.css:

```
body{

    background-image: url('/static/geral/img/bg1.png');
    background-size: cover;

}

p{
    color: var(--light-color);
}
```

Ficou faltando a imagem, adicione a imagem ‘bg1.png’ dentro de templates/static/geral/img/bg1.png

Imagens para download 👇

Acesse pelo NOTION para realizar o download das imagens, link no início do PDF:

bg1.png

bg2.png

logo.png

Importe o css de cadastro:

```
<link href="{% static 'usuarios/css/css.css' %}" rel="stylesheet">
```

Adicione a logo da empresa:

```
<img src="{% static 'geral/img/logo.png' %}" alt="">
```

Execute as migrações:

```
python manage.py makemigrations
python manage.py migrate
```

Crie as funcionalidades de cadastro na view:

```
def cadastro(request):
    if request.method == "GET":
        return render(request, 'cadastro.html')
    else:
        primeiro_nome = request.POST.get('primeiro_nome')
        ultimo_nome = request.POST.get('ultimo_nome')
        username = request.POST.get('username')
        email = request.POST.get('email')
        senha = request.POST.get('senha')
        confirmar_senha = request.POST.get('confirmar_senha')

        if not senha == confirmar_senha:
            return redirect('/usuarios/cadastro')

        if len(senha) < 6:
            return redirect('/usuarios/cadastro')

        try:
            # Username deve ser único!
            user = User.objects.create_user(
                first_name=primeiro_nome,
                last_name=ultimo_nome,
                username=username,
                email=email,
                password=senha,
            )
        except:
            return redirect('/usuarios/cadastro')


        return redirect('/usuarios/cadastro')
```

Altere o form para enviar os dados para a view:

```
<form action="{% url 'cadastro' %}" method="POST"> {% csrf_token %}
```

Configure o Django message em settings.py:

```
from django.contrib.messages import constants

MESSAGE_TAGS = {
    constants.DEBUG: 'alert-primary',
    constants.ERROR: 'alert-danger',
    constants.WARNING: 'alert-warning',
    constants.SUCCESS: 'alert-success',
    constants.INFO: 'alert-info',
}
```

Agora adicione as mensagens nos pontos estratégicos do código:

```
messages.add_message(request, constants.ERROR, 'As senhas não coincidem')
```

Exiba as mensagens no HTML de cadastro:

```
{% if messages %}
    <br>
    {% for message in messages %}
    <div class="alert {{ message.tags }}">{{ message }}</div>
    {% endfor %}
{% endif %}
```

▼ Login

Crie uma URL para login:

```
path('login/', views.logar, name="login"),
```

Crie a view logar:

```
def logar(request):
    if request.method == "GET":
        return render(request, 'login.html')
    else:
        username = request.POST.get('username')
        senha = request.POST.get('senha')

        user = authenticate(username=username, password=senha)

        if user:
            login(request, user)
            # Acontecerá um erro ao redirecionar por enquanto, resolveremos nos próximos passos
            return redirect('/')
        else:
            messages.add_message(request, constants.ERROR, 'Usuario ou senha inválidos')
            return redirect('/usuarios/login')
```

Crie o login.html:

```
{% extends "bases/base.html" %}
{% load static %}

{% block 'head' %}

    <link href="{% static 'usuarios/css/css.css' %}" rel="stylesheet">

{% endblock 'head' %}
```

```
{% block 'conteudo' %}
    <div class="container">

        <h3 class="font-destaque-secundaria"> <img width="10%" src="{% static 'geral/img/logo.png' %}" alt="">Cadastre-se</h3>

        {% if messages %}
            <br>
            {% for message in messages %}
                <div class="alert {{ message.tags }}">{{ message }}</div>
            {% endfor %}
        {% endif %}


        <div>
            <form action="{% url 'login' %}" method="POST">{% csrf_token %}
                <label>Username</label>
                <br>
                <input type="text" class="input-default" name="username">
                <br>
                <br>
                <label>Senha</label>
                <br>
                <input type="text" class="input-default" name="senha">
                <br>
                <br>
                <input type="submit" class="btn-default" value="Logar">
            </form>
        </div>
    </div>
{% endblock %}
```

▼ Exames configurações iniciais

Crie o app exames:

```
python manage.py startapp exames
```

INSTALE O APP!

Crie as models:

```
from django.db import models
from django.contrib.auth.models import User


class TiposExames(models.Model):
    tipo_choices = (
        ('I', 'Exame de imagem'),
        ('S', 'Exame de Sangue')
    )
    nome = models.CharField(max_length=50)
    tipo = models.CharField(max_length=2, choices=tipo_choices)
    preco = models.FloatField()
    disponivel = models.BooleanField(default=True)
    horario_inicial = models.IntegerField()
    horario_final = models.IntegerField()

    def __str__(self):
        return self.nome

class SolicitacaoExame(models.Model):
    choice_status = (
        ('E', 'Em análise'),
        ('F', 'Finalizado')
    )
    usuario = models.ForeignKey(User, on_delete=models.DO_NOTHING)
    exame = models.ForeignKey(TiposExames, on_delete=models.DO_NOTHING)
    status = models.CharField(max_length=2, choices=choice_status)
    resultado = models.FileField(upload_to="resultados", null=True, blank=True)
    requer_senha = models.BooleanField(default=False)
    senha = models.CharField(max_length=6, null=True, blank=True)
```

```
    def __str__(self):
        return f'{self.usuario} | {self.exame.nome}'

class PedidosExames(models.Model):
    usuario = models.ForeignKey(User, on_delete=models.DO_NOTHING)
    exames = models.ManyToManyField(SolicitacaoExame)
    agendado = models.BooleanField(default=True)
    data = models.DateField()

    def __str__(self):
        return f'{self.usuario} | {self.data}'
```

EXECUTE AS MIGRAÇÕES!

Crie um super usuário:

```
python manage.py createsuperuser
```

Cadastre as models na área administrativa:

```
from django.contrib import admin
from .models import TiposExames, PedidosExames, SolicitacaoExame

admin.site.register(TiposExames)
admin.site.register(PedidosExames)
admin.site.register(SolicitacaoExame)
```