

Tópicos de Segurança

Programador FullStack Flask

Autenticação

Autorização

Instalação do JWT

```
quickreport> ?
```

Instalação do JWT

```
quickreport> ?
```

Instalação do JWT


```
quickreport> ?
```

Instalação do JWT

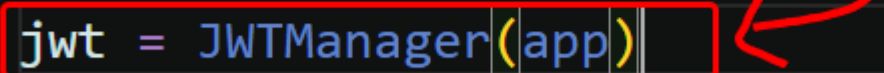
pip install Flask-JWT-Extended

Flask	3.0.3
Flask-JWT-Extended	4.6.0
Flask-Migrate	4.0.7
Flask-Script	2.0.6
Flask-SQLAlchemy	3.1.1
Flask-WTF	1.2.1
greenlet	3.0.3
itsdangerous	2.2.0
Jinja2	3.1.3
Mako	1.3.3
MarkupSafe	2.1.5
mysql-connector-python	8.3.0
mysqlclient	2.2.4
pip	24.0
PyJWT	2.8.0
SQLAlchemy	2.0.29
typing_extensions	4.11.0
Werkzeug	3.0.2
WTForms	3.1.2


```
1  #importando o flask
2  from flask import Flask
3  #importando o SQLAlchemy
4  from flask_sqlalchemy import SQLAlchemy
5  from flask_migrate import Migrate, upgrade
6  from flask_wtf import CSRFProtect
7  from flask_jwt_extended import JWTManager
```



```
1  #importando o flask
2  from flask import Flask
3  #importando o SQLAlchemy
4  from flask_sqlalchemy import SQLAlchemy
5  from flask_migrate import Migrate, upgrade
6  from flask_wtf import CSRFProtect
7  from flask_jwt_extended import JWTManager
8  #criando o aplicativo
9  app = Flask(__name__)
10 #puxando o arquivo config.py
11 app.config.from_object('config')
12 #criando um objeto db da classe SQLAlchemy
13 db = SQLAlchemy(app)
14 #criar uma variável migrate e passar a instância da aplicação e do db
15 migrate = Migrate(app, db)
16 jwt = JWTManager(app)
17 csrf = CSRFProtect(app)
18 csrf.init_app(app)
```



CONFIG.py

```
15  #connection string
16  SQLALCHEMY_DATABASE_URI=f'mysql://{USERNAME}:{PASSWORD}@{SERVER}/{DB}'
17  #modificação
18  SQLALCHEMY_TRACK_MODIFICATIONS = True
19  #chave secreta - hash (chave criptografada)
20  #entrar em qualquer site que gere hash - colocar o hash
21  #publicar.
22  SECRET_KEY ="8a4dbb9594173ae2747f9704468a89bd" ←
23
24
25
```

WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.

* Running on http://127.0.0.1:5000

Press CTRL+C to quit

* Restarting with stat

* Debugger is active!

* Debugger PIN: 498-349-401

127.0.0.1 - - [20/Jun/2024 10:04:39] "GET / HTTP/1.1" 200 -

127.0.0.1 - - [20/Jun/2024 10:04:39] "GET /static/css/bootstrap.css HTTP/1.1" 304 -

□

```
jwt = JWTManager(app)
```

Biblioteca de Gerar Tokens

No projeto piloto nosso UsuarioForm não tem senha?

```
1 from flask_wtf import FlaskForm
2 from wtforms import StringField, SelectField
3 from wtforms.validators import DataRequired, Email
4
5 class UsuarioForm(FlaskForm):
6     nome = StringField('Nome', validators=[DataRequired()])
7     email = StringField('Email', validators=[DataRequired()])
8     nivel_id = SelectField('Nivel', coerce=int, validators=[DataRequired()])
```

Nosso model não tem senha?

```
1  from app import db #SQLAlchemy - Migrate:Migrar Classe para Tabela
2  from sqlalchemy import Text
3  from sqlalchemy.orm import relationship
4
5  #Holdai(MeuPai)
6  class Usuario(db.Model):
7      __tablename__ = "usuario"
8      #id = db.Column(tipo,chave,auto)
9      id = db.Column(db.Integer,primary_key=True,autoincrement=True)
10     nome = db.Column(db.String(200))
11     email = db.Column(db.String(200))
12     fk_nivel_id= db.Column(db.Integer,db.ForeignKey('nivel.id'))
13     nivel = relationship("Nivel",back_populates="usuarios")
```

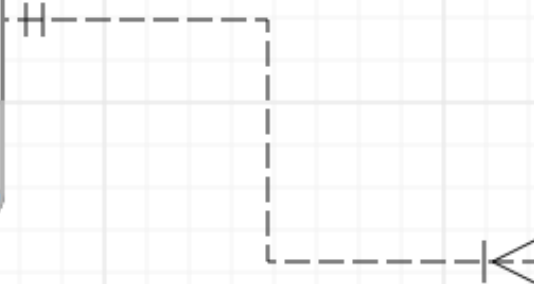
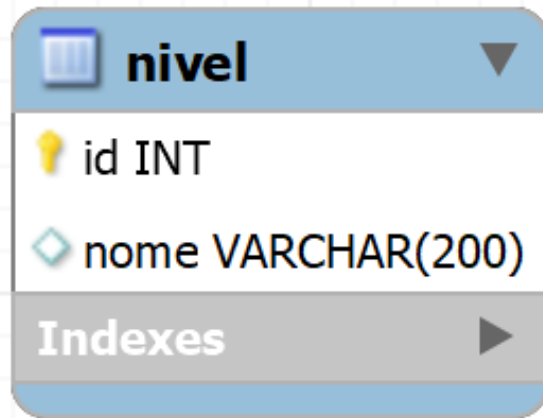
Migrate nele! Vamos?


```
1 from app import db #SQLAlchemy - Migrate:Migrar Classe para Tabela
2 from sqlalchemy import Text
3 from sqlalchemy.orm import relationship
4
5
6 class Usuario(db.Model):
7     __tablename__ = "usuario"
8     #id = db.Column(tipo,chave,auto)
9     id = db.Column(db.Integer,primary_key=True,autoincrement=True)
10    nome = db.Column(db.String(200))
11    email = db.Column(db.String(200))
12    senha = db.Column(db.String(255))
13    fk_nivel_id= db.Column(db.Integer,db.ForeignKey('nivel.id'))
14    nivel = relationship("Nivel",back_populates="usuarios")
15
16
```

```
flask db migrate -m "adicionnei senha"
```

```
INFO [alembic.runtime.migration] Context impl MySQLImpl.  
INFO [alembic.runtime.migration] Will assume non-transactional DDL.  
INFO [alembic.autogenerate.compare] Detected added column 'usuario.senha'  
Generating C:\Users\BIBLIOTECA\Desktop\projetosprof\quickreport\migrations\versions\92cdc216094c_adicionei_senha.py ... done
```

flask db upgrade



Diminuir o tamanho dos campos

```
class Usuario(db.Model):  
    __tablename__ = "usuario"  
    #id = db.Column(tipo, chave, auto)  
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)  
    nome = db.Column(db.String(80))  
    email = db.Column(db.String(80))  
    senha = db.Column(db.String(255))  
    fk_nivel_id = db.Column(db.Integer, db.ForeignKey('nivel.id'))  
    nivel = relationship("Nivel", back_populates="usuarios")
```

```
flask db migrate -m "DiminirTamanhoCampoNomeEmail"
```

al DDL.

```
m VARCHAR(length=200) to String(length=80) on 'usuario.nome'
```

```
m VARCHAR(length=200) to String(length=80) on 'usuario.email'
```

```
rt\migrations\versions\9770d2238d9c_diminirtamanhocamponomeemail.py
```

flask db upgrade

```
flask db upgrade
```

```
al DDL.
```

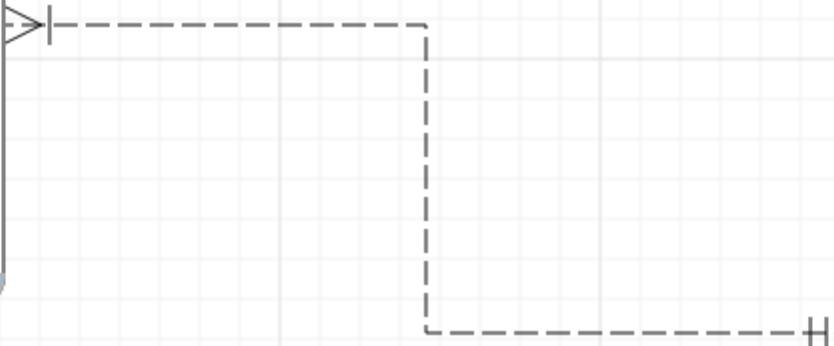
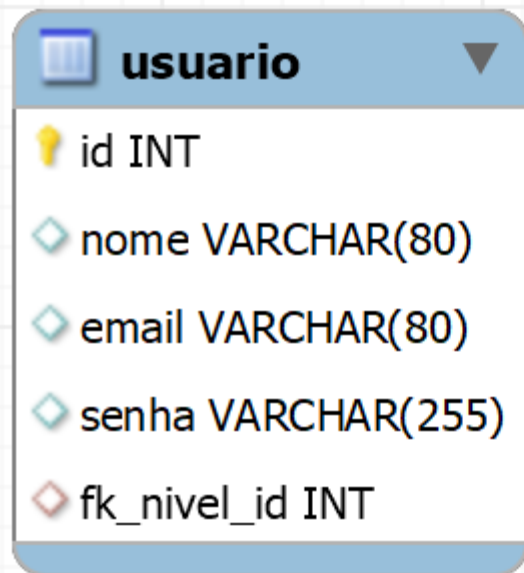
```
c -> 9770d2238d9c, DiminirTamanhoCampoNomeEmail
```


5. describe usuario;

<

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	Field	Type	Null	Key	Default	Extra
►	id	int	NO	PRI	NULL	auto_increment
	nome	varchar(80)	YES		NULL	
	email	varchar(80)	YES		NULL	
	fk_nivel_id	int	YES	MUL	NULL	
	senha	varchar(255)	YES		NULL	





usuario



id INT



nome VARCHAR(80)



email VARCHAR(80)



senha VARCHAR(255)




fk_nivel_id INT

```
if form.validate_on_submit():  
    nome = form.nome.data  
    email = form.email.data  
    senha = form.senha.data  
    nivel_id = form.nivel_id.data
```

```
usuario = Usuario(nome=nome,email=email,senha=senha,fk_nivel_id=nivel_id)
```

```
usuario = Usuario(nome=nome, email=email, senha=senha, fk_nivel_id=nivel_id)
try:
    db.session.add(usuario)
    db.session.commit()
    return redirect(url_for('listar_usuarios'))
```

```
1 from flask_wtf import FlaskForm
2 from wtforms import StringField, SelectField, PasswordField
3 from wtforms.validators import DataRequired, Email
```



```
class UsuarioForm(FlaskForm):  
    nome = StringField('Nome', validators=[DataRequired()])  
    email = StringField('Email', validators=[DataRequired()], render_kw={"type": "email"})  
    senha = PasswordField('Senha', validators=[DataRequired()])  
    nivel_id = SelectField('Nivel', coerce=int, validators=[DataRequired()])
```

	id	nome
▶	1	administrador
	2	padrão
●	NULL	NULL

	id	nome	email	senha	fk_nivel_id
▶	1	romulo cesar leite silvestre	romulo@sistemapfira.org.br	1111111	1
	2	romulo cesar leite periera	test@gmail.com	5555555	1
	3	rone	rone@teste.com.br	123	2
	4	renata	renata@teste.com.br	66666	1
	5	teste	teste@senai.com	666667777	1
●	NULL	NULL	NULL	NULL	NULL

Antes de salvar o usuário

Devemos criptografar essa senha!

O que é criptografia?

```
pip install passlib
```

Collecting passlib

Downloading passlib-1.7.4-py2.py3-none-any.whl.metadata (1.7 kB)

Downloading passlib-1.7.4-py2.py3-none-any.whl (525 kB)

525.6/525.6 kB 2.5 MB/s eta 0:00:00

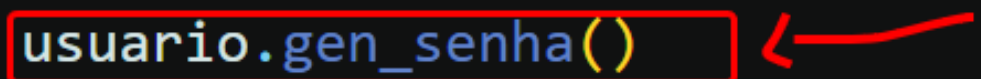
Installing collected packages: passlib

Successfully installed passlib-1.7.4

```
from passlib.hash import pbkdf2_sha256
```

```
def gen_senha(self):  
    self.senha = pbkdf2_sha256.hash(self.senha)  
  
def ver_senha(self, senha):  
    #quando fazer Login  
    #retorna true ou false  
    return pbkdf2_sha256.verify(senha, self.senha)
```

```
usuario = Usuario(nome=nome,email=email,senha=senha,fk_nivel_id=nivel_id)
usuario.gen_senha()
```




```
1 from flask import render_template, redirect, url_for
2 from app import app, db
3 from app.models.nivel_model import Nivel
4 from app.models.usuario_model import Usuario
5 from app.forms.nivel_form import NivelForm
6 from app.forms.usuario_form import UsuarioForm
7
```

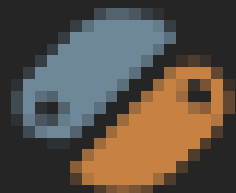
```
8
9  from sqlalchemy.orm import joinedload
10
11
12  @app.route("/cadusuario", methods=["POST", "GET"])
13  def cadastrar_usuario():
14      form = UsuarioForm()
```

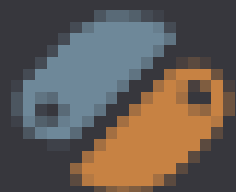
```
12 @app.route("/cadusuario", methods=["POST", "GET"])
13 def cadastrar_usuario():
14     form = UsuarioForm()
15
16     # Carregar os níveis do banco de dados
17     niveis = Nivel.query.all()
18     # Formatar os níveis para o formato necessário para o campo de sel
19     nivel_choices = [(nivel.id, nivel.nome) for nivel in niveis]
20     # Adicionar os níveis ao campo de seleção
21     form.nivel_id.choices = nivel_choices
```

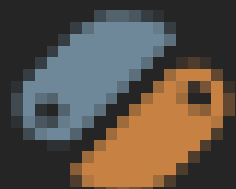
```
23     if form.validate_on_submit():
24         nome = form.nome.data
25         email = form.email.data
26         senha = form.senha.data
27         nivel_id = form.nivel_id.data
28
29         usuario = Usuario(nome=nome, email=email, senha=senha, fk_nivel_i
30         usuario.gen_senha() ←
31     try:
32         db.session.add(usuario)
33         db.session.commit()
34         return redirect(url_for('listar_usuarios'))
35     except Exception as e:
36         print(f"Erro ao cadastrar usuário:{e}")
```

```
40 @app.route("/listarusuarios", methods=["POST", "GET"])
41 def listar_usuarios():
42     #tecnicos = tecnico_model.Tecnico.query.all() # Consulta todos os
43     usuarios_com_nivel = db.session.query(Usuario).options(joinedload(
44     return render_template("usuario/lista_usuario.html", usuarios=usua
```

 curso_view.py

 escola_view.py

 login_view.py

 nivel_view.py



```
1 from app import app
```

```
2
```

```
3
```

```
4
```

```
4 @app.route("/", methods=["POST", "GET"])
```

```
5
```

```
6
```



```
4 @app.route("/", methods=["POST", "GET"])
5 def login_usuario():
6     pass
```

▼ templates

> aula_jinja

> curso

> escola


> login



> nivel

> site_estatico

> usuario

✓  login

 index.html

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, in
6      <title>Login - Projeto Piloto</title>
7  </head>
8  <body>
9
10     <h1>Tela de Login</h1>
11
12 </body>
13 </html>
```

```
1 from app import app
```

```
2 from flask import render_template
```

```
1 from app import app
2 from flask import render_template
3 @app.route("/", methods=["POST", "GET"])
4 def login_usuario():
5     return render_template("login/index.html")
```

```
28  #FIXME:view
29  from .views import nivel_view
30  from .views import usuario_view
31  from .views import escola_view
32  from .views import curso_view
33  from .views import aulajinja_view
34  from .views import login_view ←
```



127.0.0.1:5000



Tela de Login

Login dos Usuários

1. Criamos a tela de login
2. Vamos criar uma view de login
3. Um post será enviado
4. Capturar os dados
5. Verificar se os dados são válidos no banco de dados
6. Se sim gera um access token
7. E com esse hash que vamos utilizar para autenticar
8. E assim realizar requisições

Vamos começar

Tela de Login

```
<body>
  <header class="bg-dark text-white text-center py-4 mb-5">
    <div class="container">
      <h1 class="display-4">Sistema Base</h1>
      <h2>Cadastro de Usuário</h2>
    </div>
  </header>

  <div class="container">
    <form method="post">
      <div class="form-group">
        <label for="{{ form.email.id }}">Email</label>
        {{ form.email(class="form-control") }}
      </div>
      <div class="form-group">
        <label for="{{ form.senha.id }}">Senha</label>
        {{ form.senha(class="form-control") }}
      </div>
      <button class="btn btn-primary" type="submit">Logar</button>
    </form>
  </div>
</body>
```

```
1 from flask_wtf import FlaskForm
2 from wtforms import StringField, SelectField, PasswordField
3 from wtforms.validators import DataRequired, Email
4
5 class UsuarioForm(FlaskForm):
6     email = StringField('Email', validators=[DataRequired()], render_kw={"type": "email"},)
7     senha = PasswordField('Senha', validators=[DataRequired()])
8
```

```
1 from flask_wtf import FlaskForm
2 from wtforms import StringField, SelectField, PasswordField
3 from wtforms.validators import DataRequired, Email
4
5 class LoginForm(FlaskForm):
6     email = StringField('Email', validators=[DataRequired()], render_kw={"type": "email"},)
7     senha = PasswordField('Senha', validators=[DataRequired()])
8
```



flask-snippets

v0.1.3

cstrap



498,292

★★★★★ (4)

Flask snippets

Disable



Uninstall



Sistema Base

Cadastro de Usuário

Email

Senha

Logar



```
if usuario_bd and usuario_bd.ver_senha(senha):  
    access_token = create_access_token(  
        identity=usuario_bd.id,  
        expires_delta=timedelta(seconds=100)  
    )  
  
    return make_response(jsonify({  
        'access_token': access_token,  
        'message': 'Login realizado com sucesso'  
    })), 200)
```

Sistema Base

Entre em nosso Sistema

Email

Senha

Login realizado com sucesso!

Sistema Base

Entre em nosso Sistema

Email

Senha

Autenticação não sucedida. Verifique seu email e senha.

Sistema Base

Entre em nosso Sistema

Email

Senha

Logar

Login realizado com sucesso!

```
from app import app
```

```
from flask import render_template, flash, redirect, url_for
```

```
from app.forms.login_form import LoginForm
```

```
from app.models.usuario_model import Usuario
```

```
from flask_jwt_extended import create_access_token #para criar o access token
```

```
from datetime import timedelta #tempo de validade do access token
```



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

```
def logar_usuario():  
    form_login = LoginForm()  
    if form_login.validate_on_submit():  
        email = form_login.email.data  
        senha = form_login.senha.data
```

```
def consultar_email(email):  
    usuario = Usuario.query.filter_by(email=email).first() # Consulta todos os registros na  
    return usuario
```

```
usuario_banco = consultar_email(email)
```

```
if usuario_banco and usuario_banco.ver_senha(senha):
```

```
access_token = create_access_token(  
    identity=usuario_banco.id,  
    expires_delta=timedelta(seconds=100)  
)
```

```
flash('Login realizado com sucesso!', 'success') # Mensagem de sucesso
```

```
return render_template("login/index.html", form_login=form_login)
```



```
if usuario_banco and usuario_banco.ver_senha(senha):  
    access_token = create_access_token(  
        identity=usuario_banco.id,  
        expires_delta=timedelta(seconds=100)  
    )  
    flash('Login realizado com sucesso!', 'success') # Mensagem de sucesso  
    return render_template("login/index.html", form_login=form_login)
```

```
    else:  
        flash('Autenticação não sucedida. Verifique seu email e senha.', 'danger') #  
return render_template("login/index.html", form_login=form_login)
```

```
1 from app import app
2 from flask import render_template, flash, redirect, url_for
3 from app.forms.login_form import LoginForm
4 from app.models.usuario_model import Usuario
5 from flask_jwt_extended import create_access_token #para criar o access token
6 from datetime import timedelta #tempo de validade do access token
```

```
8 @app.route("/", methods=["POST", "GET"])
9 def login_usuario():
10     form_login = LoginForm()
11     if form_login.validate_on_submit():
12         email = form_login.email.data
13         senha = form_login.senha.data
14         usuario_banco = consultar_email(email)
15
16         if usuario_banco and usuario_banco.ver_senha(senha):
17             access_token = create_access_token(
18                 identity=usuario_banco.id,
19                 expires_delta=timedelta(seconds=100)
20             )
21             flash('Login realizado com sucesso!', 'success') # Mensagem de sucesso
22             return render_template("login/index.html", form_login=form_login)
23
24         else:
25             flash('Autenticação não sucedida. Verifique seu email e senha.', 'danger') # Mensagem de falha
26
27     return render_template("login/index.html", form_login=form_login)
```

```
29 def consultar_email(email):
30     usuario = Usuario.query.filter_by(email=email).first() # Consulta todos os registros na tabela Nivel
31     return usuario
32
33
```

Sistema Base

Entre em nosso Sistema

Email

Senha



127.0.0.1:5000

Login realizado com sucesso!

Sistema Base

Entre em nosso Sistema

Email

Senha

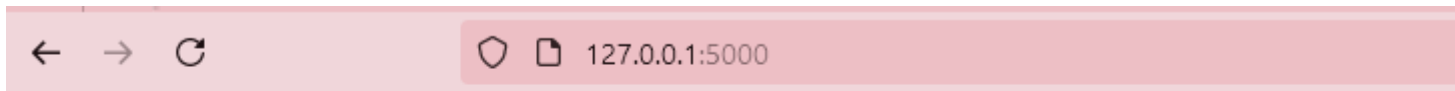
Logar

Sistema Base

Entre em nosso Sistema

Email

Senha



Você não está autorizado acessar o sistema

Sistema Base

Entre em nosso Sistema

Email

Senha

Logar

Sistema Base

Entre em nosso Sistema

Email

Senha



🔒 📄 127.0.0.1:5000

Login realizado com sucesso! Bem-Vindo ao projeto piloto

Sistema Base

Entre em nosso Sistema

Email	<input type="text" value="rex@gmail.com"/>
Senha	<input type="password" value="●●●"/>
<input type="button" value="Logar"/>	



🛡️ 📄 127.0.0.1:5000

Autenticação não sucedida. Projeto piloto não acessado. Verifique seu email e senha.

```
if usuario_banco and usuario_banco.ver_senha(senha):
    access_token = create_access_token(
        identity=usuario_banco.id,
        expires_delta=timedelta(seconds=100)
    )
    flash('Login realizado com sucesso! Bem-Vindo ao projeto piloto', 'success') # Mensagem de sucesso
    return render_template("sucesso/index.html")

else:
    flash('Autenticação não sucedida. Projeto piloto não acessado. Verifique seu email e senha.', 'danger')
    return render_template("sucesso/erro.html")
```