Gerenciador de tarefas

Guilherme Lopes e Sara Stephanie

Universidade do Estado da Bahia

16 de junho de 2024

Sumario

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Sumario

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Introducao

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Introducao: Objetivo e Metodologia

Criar um Gerenciador de tarefas que forneca informacões que circulam entre a administracao de uma empresa e os funcionarios. Fornecendo relatórios das tarefas.

Para o embasamento teórico e uma melhor compreensao do tema, foram utilizadas vídeos aulas e pesquisas acadêmicas que pudessem fomentar o trabalho. Aumentando as fontes e elevando o nível do projeto final.

Introducao: Motivacao e Justificativa

Pensando no futuro, onde se deseja um ambiente produtivo e organizado, teve-se como ideia, a criacao de um programa em python que fosse capaz gerenciar tarefas basicas de uma obra da construcao civil, buscando proporcionar uma maior organizacao nas tarefas que deverao ser executadas.

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Referencial Teórico

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Referencial Teórico: Bibliotecas


FastAPI

Uvicorn

Requests


Pydantic

PySimpleGUI


SQLite

Proposta do programa

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Evolucao do programa

Antes o programa era capaz apenas de criar tarefas localmente sem que fosse possível que outras pessoas tivessem acesso a essa tarefa. Agora é possível que o administrador crie as tarefas e delegue essas tarefas aos funcionarios, que também conseguem visualizar essas atividades e criar relatórios as mesmas.

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Proposta do programa: Funcionamento do programa

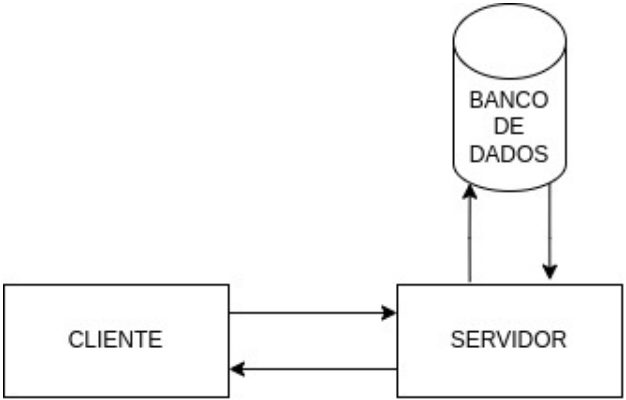


Figura: Funcionamento do programa

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Banco de dados

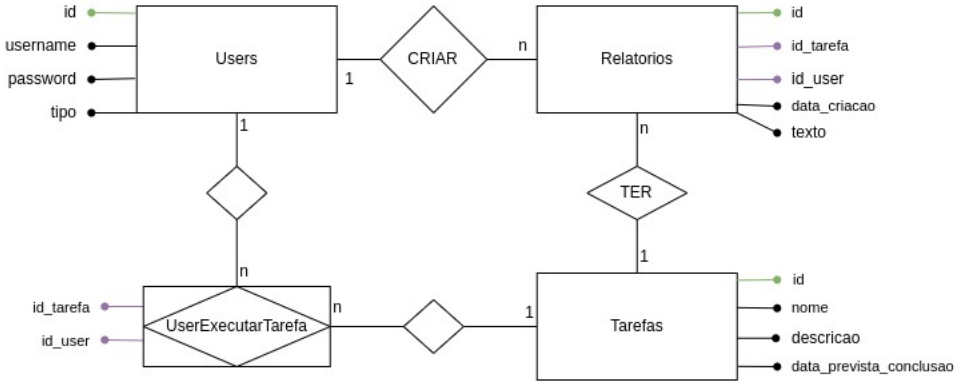
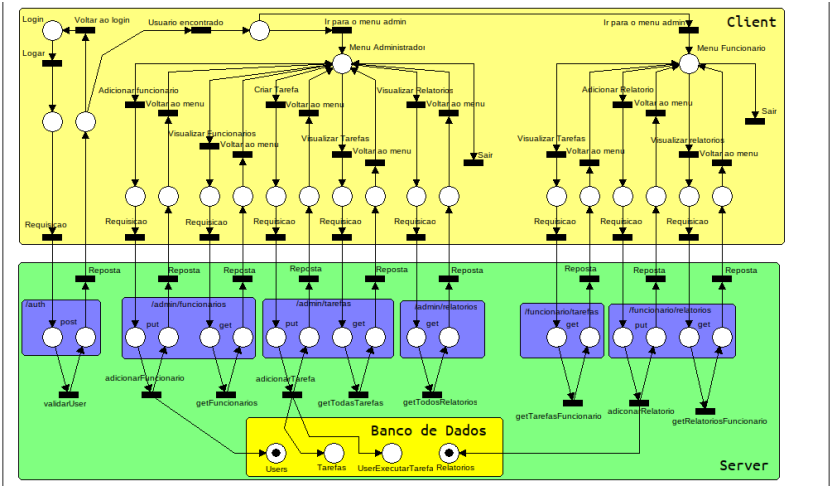


Figura: Diagrama entidade relacionamento

Proposta do programa: Rede de Petri Ordinaria

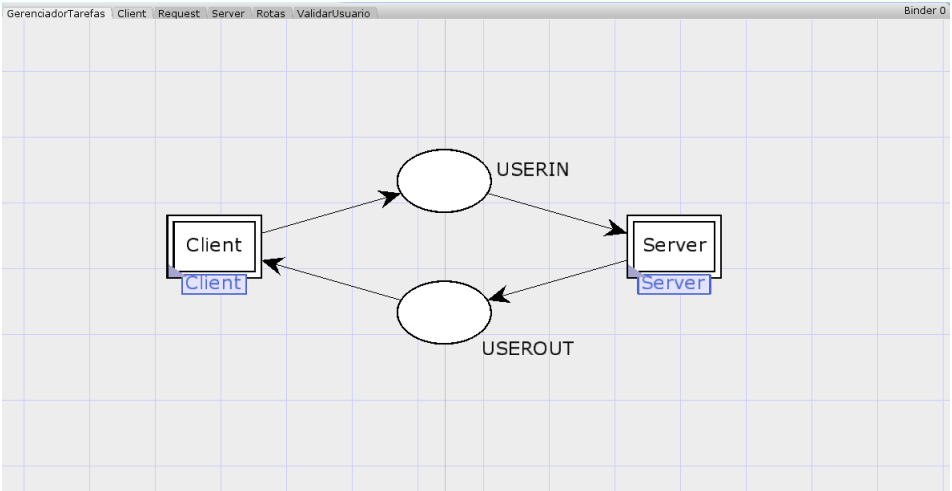


Redes de Petri: Rede de Petri Colorida

- Declarations
  - Standard priorities
  - Standard declarations
    - colset UNIT
    - colset BOOL
    - colset INT
    - colset INTINF
    - colset TIME
    - colset REAL
    - colset STRING
    - colset Login
    - colset USERIN
    - colset USEROUT=product INT * STRING * STRING * STRING;
    - colset USERS
    - colset ADMIN=product INT * STRING * STRING;
    - colset FUNC=product INT* STRING * STRING;
    - var username: STRING;
    - var username1: STRING;
    - var password: STRING;
    - var password1: STRING;
    - var tipo: STRING;
    - var id: INT;
    - var token: STRING;
  - Monitors
  - GerenciadorTarefas
    - Client
      - Request
    - Server
      - Rotas
      - ValidarUsuario

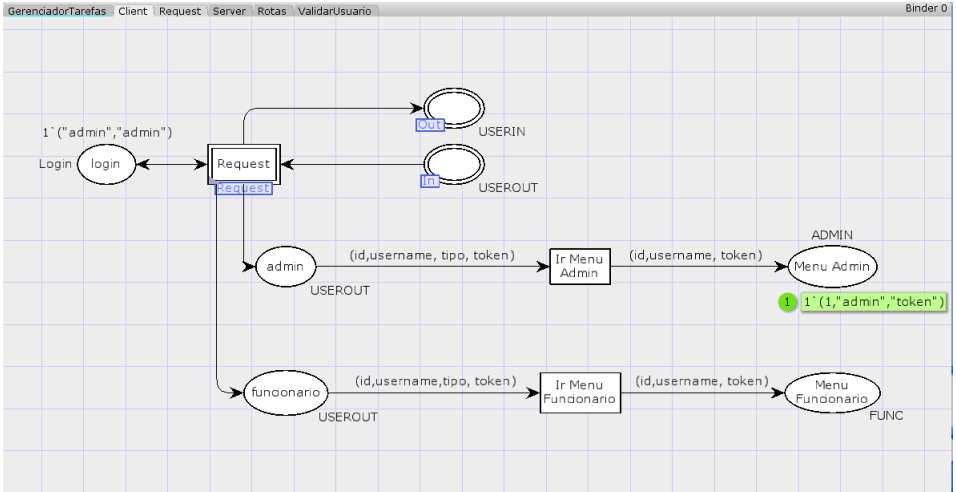
UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Redes de Petri: Rede de Petri Colorida

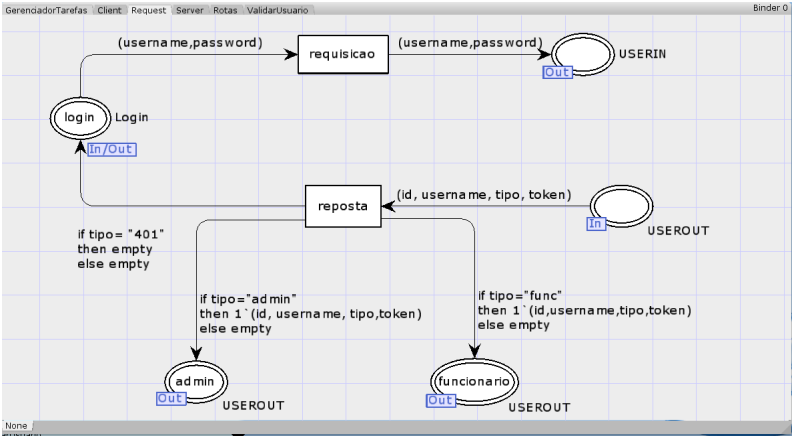


UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

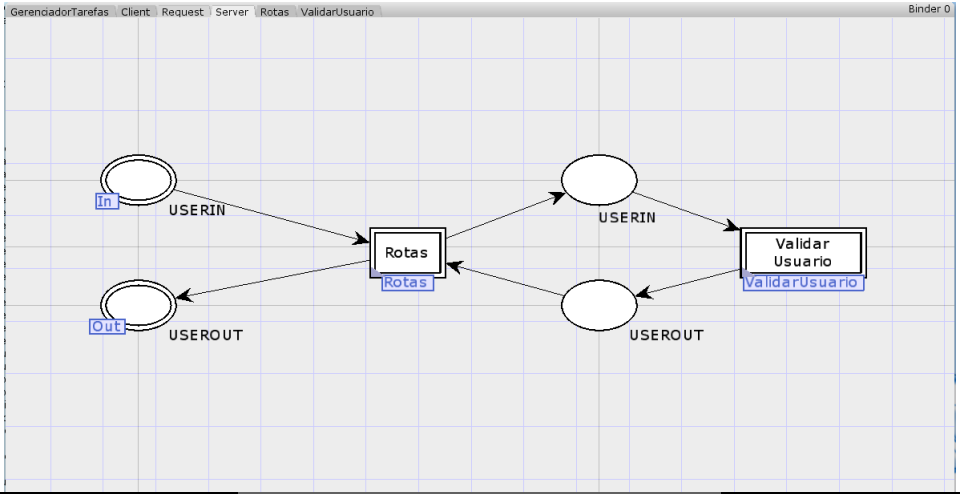
Redes de Petri: Rede de Petri Colorida



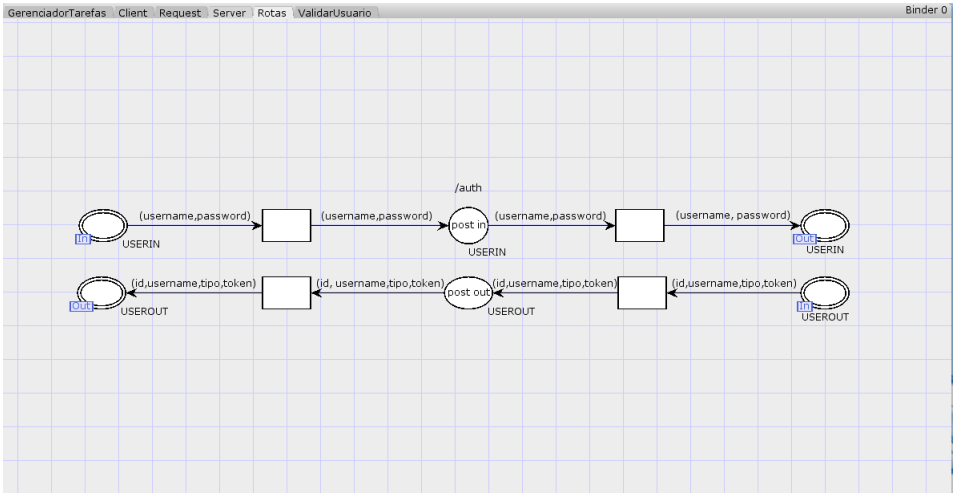
Redes de Petri: Rede de Petri Colorida



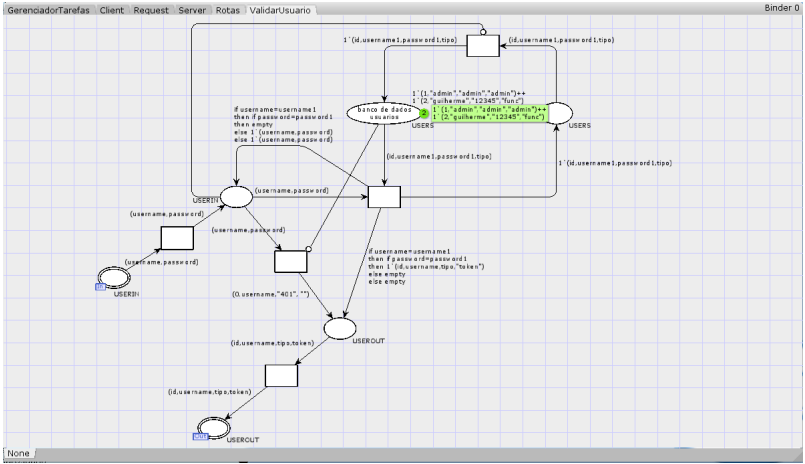
Redes de Petri: Rede de Petri Colorida



Redes de Petri: Rede de Petri Colorida



Redes de Petri: Rede de Petri Colorida



Interface Grafica

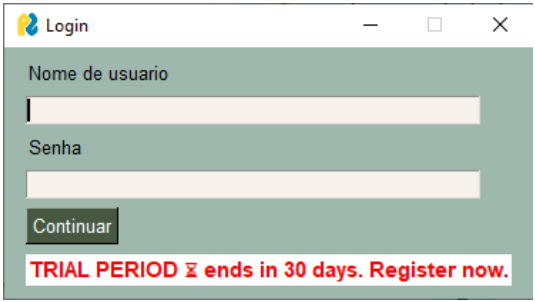
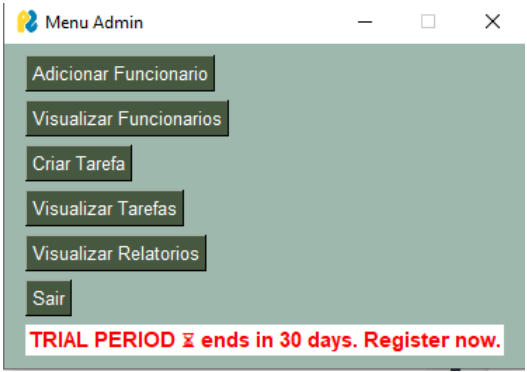


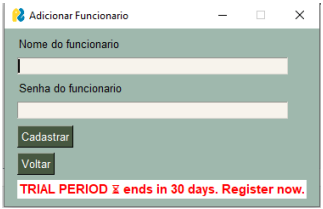
Figura: Login

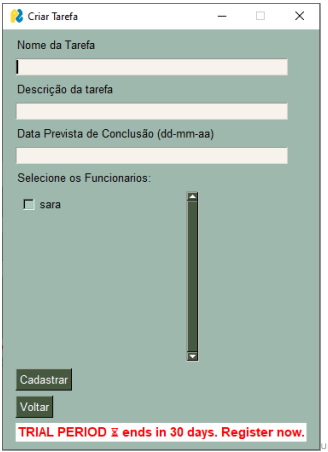
Interface Grafica

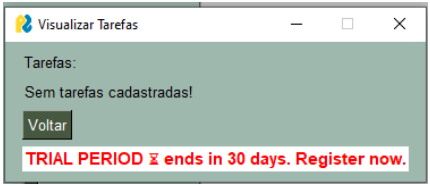


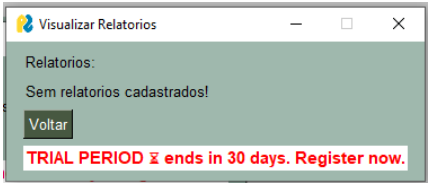
UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Interface Grafica



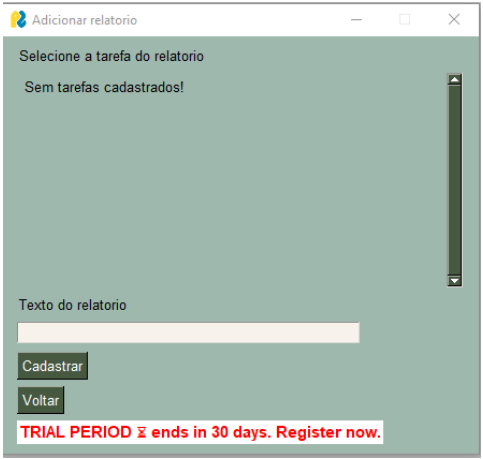






UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Interface Grafica

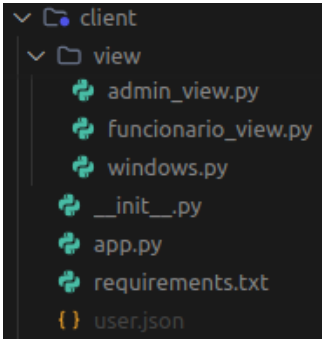


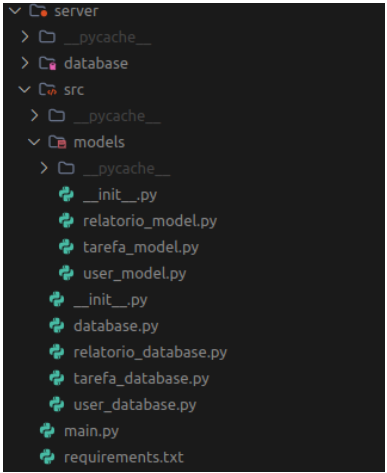
UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Codigo do Programa

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Estrutura do Projeto





UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Servidor: main

```python
from fastapi import FastAPI, HTTPException, status
import uvicorn
import jwt
from datetime import datetime, timedelta, timezone
from contextlib import asynccontextmanager
from src import database, user_database, tarefa_database, relatorio_database
from src.models.user_model import UserIn
from src.models.tarefa_model import Tarefa
from src.models.relatorio_model import Relatorio

app = FastAPI()

token_secret = "12345"

# inicializa o banco de dados de forma assincrona
@asynccontextmanager
async def lifespan(app: FastAPI):
    await database.inicializarBancoDados()
    yield

# Executa a funcao lifespan junto com a inicializacao do servidor
app.router.lifespan_context = lifespan

# Rota de autenticao. Recebe usuario e senha do client e valida se usuario esta cadastrado no banco de dados
```

Servidor: main

```python
@app.post("/auth")
async def autenticar_user(user: UserIn):
    validarUser = user_database.validarUser(user=user)

    if validarUser.get("status"):
        token = jwt.encode({
            'exp': datetime.now(timezone.utc) + timedelta(minutes=30),
            "nivel-acesso": validarUser.get("user").tipo,
            "id": validarUser.get("user").id
        }, key=token_secret, algorithm='HS256')

        validarUser.get("user").token = token

        return validarUser.get("user")
    else:
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Usuario nao encontrado!"
        )

# Rota para adicionar funcionario, recebe nome de usuario e senha
@app.put("/admin/funcionarios")
async def adicionar_funcionario(func: UserIn, token: str):
```

Servidor: main

```
48      try:
49          token_deco = jwt.decode(token, key=token_secret, algorithms="HS256")
50
51          if token_deco.get("nivel-acesso") == 'admin':
52              if user_database.adicionarFuncionario(func=func):
53                  return "Funcionario Adicionado com Sucesso!"
54              else:
55                  raise HTTPException(
56                      status_code=status.HTTP_501_NOT_IMPLEMENTED,
57                      detail="Erro ao adicionar novo funcionario!"
58                  )
59          else:
60              raise PermissionError
61      except jwt.DecodeError:
62          return HTTPException(
63              status_code=status.HTTP_401_UNAUTHORIZED,
64              detail="Token invalido!"
65          )
66      except jwt.ExpiredSignatureError:
67          return HTTPException(
68              status_code=status.HTTP_401_UNAUTHORIZED,
69              detail="Token Expirado!"
70          )
```

Servidor: main

```
71      except PermissionError:
72          return HTTPException(
73              status_code=status.HTTP_401_UNAUTHORIZED,
74              detail="Permissao Negada!"
75          )
76
77  # Rota que lista todos os funcionario cadastrados no banco de dados
78  @app.get("/admin/funcionarios")
79  async def lista_funcionarios(token: str):
80      try:
81          token_deco = jwt.decode(token, key=token_secret, algorithms="HS256")
82
83          if token_deco.get("nivel-acesso") == 'admin':
84              return user_database.getFuncionarios()
85          else:
86              raise PermissionError
87      except jwt.DecodeError:
88          return HTTPException(
89              status_code=status.HTTP_401_UNAUTHORIZED,
90              detail="Token invalido!"
91          )
92      except jwt.ExpiredSignatureError:
93          return HTTPException(
```

Servidor: main

```
94              status_code=status.HTTP_401_UNAUTHORIZED,
95              detail="Token Expirado!"
96          )
97      except PermissionError:
98          return HTTPException(
99              status_code=status.HTTP_401_UNAUTHORIZED,
100             detail="Permissao Negada!"
101         )
102
103 # Rota que adiciona uma nova tarefa do banco de dados. Recebe nome, descricao, data prevista de conclusao e id
        dos funcionarios
104 @app.put("/admin/tarefas")
105 async def adicionar_tarefa(tarefa: Tarefa, token: str):
106     try:
107         token_deco = jwt.decode(token, key=token_secret, algorithms="HS256")
108
109         if token_deco.get("nivel-acesso") == 'admin':
110             if tarefa_database.adicionarTarefa(tarefa=tarefa):
111                 return "Tarefa adicionada com Sucesso!"
112             else:
113                 raise HTTPException(
114                     status_code=status.HTTP_501_NOT_IMPLEMENTED,
115                     detail="Erro ao adicionar tarefa!"
```

Servidor: main

```
116                 )
117         else:
118             raise PermissionError
119     except jwt.DecodeError:
120         return HTTPException(
121             status_code=status.HTTP_401_UNAUTHORIZED,
122             detail="Token invalido!"
123         )
124     except jwt.ExpiredSignatureError:
125         return HTTPException(
126             status_code=status.HTTP_401_UNAUTHORIZED,
127             detail="Token Expirado!"
128         )
129     except PermissionError:
130         return HTTPException(
131             status_code=status.HTTP_401_UNAUTHORIZED,
132             detail="Permissao Negada!"
133         )
134
135 # Rota que lista todas as tarefas cadastradas no sistema
136 @app.get("/admin/tarefas")
137 async def get_todas_tarefas(token: str):
138     try:
```

Servidor: main

```
139         token_deco = jwt.decode(token, key=token_secret, algorithms="HS256")
140
141         if token_deco.get("nivel-acesso") == 'admin':
142             return tarefa_database.getTodasTarefas()
143         else:
144             raise PermissionError
145     except jwt.DecodeError:
146         return HTTPException(
147             status_code=status.HTTP_401_UNAUTHORIZED,
148             detail="Token invalido!"
149         )
150     except jwt.ExpiredSignatureError:
151         return HTTPException(
152             status_code=status.HTTP_401_UNAUTHORIZED,
153             detail="Token Expirado!"
154         )
155     except PermissionError:
156         return HTTPException(
157             status_code=status.HTTP_401_UNAUTHORIZED,
158             detail="Permissao Negada!"
159         )
160
161 # Rota que lista todos os relatorios cadastrados no sistema
```

Servidor: main

```python
@app.get("/admin/relatorios")
async def get_todos_relatorios(token: str):
    try:
        token_deco = jwt.decode(token, key=token_secret, algorithms="HS256")

        if token_deco.get("nivel-acesso") == 'admin':
            return relatorio_database.getTodosRelatorios()
        else:
            raise PermissionError
    except jwt.DecodeError:
        return HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Token invalido!"
        )
    except jwt.ExpiredSignatureError:
        return HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Token Expirado!"
        )
    except PermissionError:
        return HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Permissao Negada!"
```

UNEB
IVERSIDADE DO ESTADO DA BAHIA

Servidor: main

```
185         )
186
187 # Rota que lista todas as tarefas do funcionario. Recebe o id do usuario
188 @app.get("/funcionario/tarefas")
189 async def get_funcionario_tarefas(token: str):
190     try:
191         token_deco = jwt.decode(token, key=token_secret, algorithms="HS256")
192
193         if token_deco.get("nivel-acesso") == 'funcionario':
194             return tarefa_database.getTarefasFuncionario(idUser=token_deco.get("id"))
195         else:
196             raise PermissionError
197     except jwt.DecodeError:
198         return HTTPException(
199             status_code=status.HTTP_401_UNAUTHORIZED,
200             detail="Token invalido!"
201         )
202     except jwt.ExpiredSignatureError:
203         return HTTPException(
204             status_code=status.HTTP_401_UNAUTHORIZED,
205             detail="Token Expirado!"
206         )
207     except PermissionError:
```

Servidor: main

```
208         return HTTPException(
209             status_code=status.HTTP_401_UNAUTHORIZED,
210             detail="Permissao Negada!"
211         )
212
213 # Rota que adiciona um novo relatorio. Recebe id do usuario, id da tarefa, texto, data de criacao
214 @app.put("/funcionario/relatorios")
215 async def adicionar_relatorio(relatorio: Relatorio, token: str):
216     try:
217         token_deco = jwt.decode(token, key=token_secret, algorithms="HS256")
218
219         if token_deco.get("nivel-acesso") == 'funcionario':
220             if relatorio_database.adicionarRelatorio(relatorio=relatorio):
221                 return "Relatorio adicionado com sucesso!"
222             else:
223                 raise HTTPException(
224                     status_code=status.HTTP_501_NOT_IMPLEMENTED,
225                     detail="Erro ao adicionar tarefa!"
226                 )
227         else:
228             raise PermissionError
229     except jwt.DecodeError:
230         return HTTPException(
```

Servidor: main

```
231             status_code=status.HTTP_401_UNAUTHORIZED,
232             detail="Token invalido!"
233         )
234     except jwt.ExpiredSignatureError:
235         return HTTPException(
236             status_code=status.HTTP_401_UNAUTHORIZED,
237             detail="Token Expirado!"
238         )
239     except PermissionError:
240         return HTTPException(
241             status_code=status.HTTP_401_UNAUTHORIZED,
242             detail="Permissao Negada!"
243         )
244
245 # Rota que lista todos relatorios cadastrados pelo funcionario
246 @app.get("/funcionario/relatorios")
247 async def relatorios_cadastrados(token: str):
248     try:
249         token_deco = jwt.decode(token, key=token_secret, algorithms="HS256")
250
251         if token_deco.get("nivel-acesso") == 'funcionario':
252             return relatorio_database.getRelatoriosFuncionario(idUser=token_deco.get("id"))
253         else:
```

Servidor: main

```
254             raise PermissionError
255     except jwt.DecodeError:
256         return HTTPException(
257             status_code=status.HTTP_401_UNAUTHORIZED,
258             detail="Token invalido!"
259         )
260     except jwt.ExpiredSignatureError:
261         return HTTPException(
262             status_code=status.HTTP_401_UNAUTHORIZED,
263             detail="Token Expirado!"
264         )
265     except PermissionError:
266         return HTTPException(
267             status_code=status.HTTP_401_UNAUTHORIZED,
268             detail="Permissao Negada!"
269         )
270
271 # Inicializa o servidor se o arquivo for o principal
272 if __name__ == "__main__":
273     uvicorn.run(app=app)
```

Codigo do Programa: database

```python
import sqlite3

# Cria a conexao com banco de dados
def conexaoBancoDados() -> sqlite3.Connection:

    caminhoBd = "./server/database/gerenciador-tarefas.db"

    conn = sqlite3.connect(caminhoBd)

    return conn

# Inicializa banco de dados
async def inicializarBancoDados() -> None:
    conn = conexaoBancoDados()
    cursor = conn.cursor()

    cursor.execute('''CREATE TABLE IF NOT EXISTS Users(
                   id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
                   username TEXT NOT NULL UNIQUE,
                   password TEXT,
                   tipo TEXT CHECK(tipo IN ('admin', 'funcionario')) NOT NULL DEFAULT 'funcionario'
    );''')

    cursor.execute('''CREATE TABLE IF NOT EXISTS Tarefas(
```

Codigo do Programa: database

```
25                  id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
26                  nome TEXT NOT NULL UNIQUE,
27                  descricao TEXT NOT NULL,
28                  data_prevista_conclusao TEXT
29      );''')
30
31      cursor.execute('''CREATE TABLE IF NOT EXISTS Relatorios(
32                  id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
33                  id_user INTEGER NOT NULL,
34                  id_tarefa INTEGER NOT NULL,
35                  data_criacao TEXT NOT NULL,
36                  texto TEXT,
37                  FOREIGN KEY(id_user) REFERENCES Users(id),
38                  FOREIGN KEY(id_tarefa) REFERENCES Tarefas(id)
39      );''')
40
41      cursor.execute('''CREATE TABLE IF NOT EXISTS UserExecutarTarefa(
42                  id_user INTEGER NOT NULL,
43                  id_tarefa INTEGER NOT NULL,
44                  FOREIGN KEY(id_user) REFERENCES Users(id),
45                  FOREIGN KEY(id_tarefa) REFERENCES Tarefas(id)
46      );''')
47
```

Codigo do Programa: database

```
48      cursor.execute("SELECT * FROM Users WHERE username = 'admin';")
49      query = cursor.fetchone()
50
51      if query == None:
52          cursor.execute('''INSERT INTO Users (username, password, tipo) VALUES ('admin', 'admin', 'admin');''')
53
54      conn.commit()
55      print("Conexao com banco de dados bem sucedida!")
56      conn.close()
57
```

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Servidor: tarefa_database

```python
from src.models.tarefa_model import Tarefa
from src.database import conexaoBancoDados

# Adiciona uma nova tarefa no banco de dados. Recebe como argumento um objeto Tarefa
def adicionarTarefa(tarefa: Tarefa) -> bool:
    try:
        conn = conexaoBancoDados()
        cursor = conn.cursor()

        cursor.execute("""INSERT INTO Tarefas
                    (nome, descricao, data_prevista_conclusao) VALUES
                    (?, ?, ?);
                    """, (tarefa.nome, tarefa.descricao, tarefa.data_prevista_conclusao))

        cursor.execute("""SELECT id FROM Tarefas
                        WHERE nome=?;""", (tarefa.nome,))

        id_tarefa = cursor.fetchone()[0]

        for func in tarefa.funcionariosId:
            cursor.execute("""INSERT INTO UserExecutarTarefa
                        (id_user, id_tarefa) VALUES
                        (?, ?);""", (func, id_tarefa))

```

Servidor: tarefa_database

```
25          conn.commit()
26          conn.close()
27          return True
28      except Exception as err:
29          print(err)
30          return False
31  # Retorna uma lista de dicionarios com todas as tarefas cadastradas no banco de dados
32  def getTodasTarefas() -> list:
33      conn = conexaoBancoDados()
34      cursor = conn.cursor()
35      tarefasId = {}
36      tarefas = []
37      cursor.execute("""SELECT * FROM UserExecutarTarefa;""")
38      relacaoUserTarefa = cursor.fetchall()
39
40      for rel in relacaoUserTarefa:
41          if rel[1] in tarefasId:
42              tarefasId[rel[1]].append(rel[0])
43          else:
44              tarefasId[rel[1]] = [rel[0]]
45      c = 0
46      for key, value in tarefasId.items():
47          cursor.execute("""SELECT nome, descricao, data_prevista_conclusao FROM Tarefas
```

Servidor: tarefa_database

```
48                          WHERE id = ?;""", (key,))
49          tarefa = cursor.fetchone()
50
51          tarefas.append({
52              "id": key,
53              "nome": tarefa[0],
54              "descricao": tarefa[1],
55              "data_prevista_conclusao": tarefa[2],
56              "funcionarios": []
57          })
58
59          for funcId in value:
60              cursor.execute("""SELECT username FROM Users
61                              WHERE id = ?;""", (funcId,))
62              funcName = cursor.fetchone()[0]
63              tarefas[c]["funcionarios"].append(funcName)
64          c += 1
65      conn.close()
66      return tarefas
67  # Retorna uma lista de dicionario com todas as tarefas do funcionario. Recebe o id do funcionario como
        argumento
68  def getTarefasFuncionario(idUser: int) -> list:
69      conn = conexaoBancoDados()
```

Servidor: tarefa_database

```
70      cursor = conn.cursor()
71      cursor.execute("""SELECT id_tarefa FROM UserExecutarTarefa
72                    WHERE id_user = ?;""", (idUser,))
73      idTarefas = cursor.fetchall()
74      tarefas = []
75      c = 0
76      while c < len(idTarefas):
77          cursor.execute("""SELECT * FROM Tarefas
78                        WHERE id = ?;""", (idTarefas[c][0],))
79          tarefa = cursor.fetchall()
80          tarefas.append({
81              "id": tarefa[0][0],
82              "nome": tarefa[0][1],
83              "descricao": tarefa[0][2],
84              "data_prevista_conclusao": tarefa[0][3]
85          })
86          c += 1
87      conn.close()
88      return tarefas
```

Servidor: relatorio_database

```python
from src.database import conexaoBancoDados
from src.models.relatorio_model import Relatorio

# Adiciona um novo relatorio no banco de dados. Recebe um objeto Relatorio como argumento
def adicionarRelatorio(relatorio: Relatorio) -> bool:
    try:
        conn = conexaoBancoDados()
        cursor = conn.cursor()

        cursor.execute("""INSERT INTO Relatorios
                    (id_user, id_tarefa, data_criacao, texto) VALUES
                    (?, ?, ?, ?);
                    """, (relatorio.id_user, relatorio.id_tarefa, relatorio.data_criacao, relatorio.texto))

        conn.commit()
        conn.close()

        return True
    except Exception as err:
        print(err)
        return False

# Retorna uma lista de dicionarios com os relatorios do funcionario. Recebe o id do funcionario como argumento
def getRelatoriosFuncionario(idUser: int) -> list:
```

Servidor: relatorio_database

```
25      conn = conexaoBancoDados()
26      cursor = conn.cursor()
27
28      cursor.execute("""SELECT id, id_tarefa, data_criacao, texto FROM Relatorios
29                  WHERE id_user = ?;""", (idUser,))
30
31      relatorios = []
32      queryRelatorios = cursor.fetchall()
33
34      for relatorio in queryRelatorios:
35          cursor.execute("""SELECT nome FROM Tarefas
36                      WHERE id = ?;""", (relatorio[1],))
37          nomeTarefa = cursor.fetchone()[0]
38
39          relatorios.append({
40              "id": relatorio[0],
41              "id_tarefa": relatorio[1],
42              "nome_tarefa": nomeTarefa,
43              "data_criacao": relatorio[2],
44              "texto": relatorio[3]
45          })
46
47      conn.close()
```

Servidor: relatorio_database

```
48
49      return relatorios
50
51  # Retorna uma lista de dicionarios com todos os relatorios cadastrados no sistema
52  def getTodosRelatorios() -> list:
53      conn = conexaoBancoDados()
54      cursor = conn.cursor()
55
56      cursor.execute("""SELECT id, id_user, id_tarefa, data_criacao, texto FROM Relatorios;""")
57      query = cursor.fetchall()
58
59      relatorios = []
60
61      for relatorio in query:
62          cursor.execute("""SELECT username FROM Users
63                            WHERE id = ?;""", (relatorio[1],))
64          nomeFunc = cursor.fetchone()[0]
65
66          cursor.execute("""SELECT nome FROM Tarefas
67                            WHERE id = ?;""", (relatorio[2],))
68          nomeTarefa = cursor.fetchone()[0]
69
70          relatorios.append({
```

Servidor: relatorio_database

```
71              "id": relatorio[0],
72              "id_user": relatorio[1],
73              "id_tarefa": relatorio[2],
74              "nome_funcionario": nomeFunc,
75              "nome_tarefa": nomeTarefa,
76              "data_criacao": relatorio[3],
77              "texto": relatorio[4]
78          })
79
80      conn.close()
81
82      return relatorios
```

Servidor: user_database

```python
from src.database import conexaoBancoDados
from src.models.user_model import UserIn, UserOut

# Valida o usuario no banco de dados. Recebe como argumento um objeto UserIn
def validarUser(user: UserIn) -> bool:
    conn = conexaoBancoDados()
    cursor = conn.cursor()

    cursor.execute("""SELECT id, username, password, tipo FROM Users
                   WHERE username = ? AND password = ?;""", (user.username, user.password))

    query = cursor.fetchone()
    conn.close()

    if query != None:
        return {
            "status": True,
            "user": UserOut(id=query[0], username=query[1], tipo=query[3])}

    return {
            "status": False
        }

# Adiciona um funcionario no banco de dados. Recebe um objeto UserIn
```

UNEB
IVERSIDADE DO
TADO DA BAHIA

Servidor: user_database

```
25 def adicionarFuncionario(func: UserIn) -> bool:
26     try:
27         conn = conexaoBancoDados()
28         cursor = conn.cursor()
29 
30         cursor.execute("""INSERT INTO Users (username, password, tipo) VALUES
31                     (?, ?, 'funcionario');""", (func.username, func.password))
32 
33         conn.commit()
34         conn.close()
35         return True
36     except Exception as err:
37         print(err)
38         return False
39 
40 # Retorna uma lista de dicionarios com todos os funcionarios
41 def getFuncionarios() -> list:
42     conn = conexaoBancoDados()
43     cursor = conn.cursor()
44     cursor.execute("""SELECT id, username FROM Users
45                   WHERE tipo='funcionario';""")
46 
47     query = cursor.fetchall()
```

Servidor: user_database

```
48      conn.close()
49      funcionarios = []
50      for func in query:
51          funcionarios.append({
52                  "id": func[0],
53                  "nome": func[1]
54              })
55      return funcionarios
56
```

Servidor: user_model, tarefa_model, relatorio_model

Listing: user_model.py

```python
from pydantic import BaseModel

# Classe que gerencia a entrada dos dados do usuario
class UserIn(BaseModel):
username: str
password: str

# Classe que gerencia a saida dos dados do usuario
class UserOut(BaseModel):
id: int
username: str
tipo: str

```

Listing: tarefa_model.py

```python
from pydantic import BaseModel

#Classe que gerencia a entrada de uma nova tarefa
class Tarefa(BaseModel):
nome: str
descricao: str
data_prevista_conclusao: str
funcionariosId: list


```

Listing: relatorio_model.py

```python
from pydantic import BaseModel

# Classe que gerencia a entrada de um novo relatorio
class Relatorio(BaseModel):
  id_user: int
  id_tarefa: int
  texto: str
  data_criacao: str


```

Cliente: app

```python
import requests
import PySimpleGUI as sg
import json
from view import funcionario_view
from view import admin_view
from view import windows

url = "http://127.0.0.1:8000"

def main():
    window_login = windows.login_window()
    window_menu_admin = None
    window_menu_func = None
    window_admin_adicionar_func = None
    window_admin_visualizar_func = None
    window_admin_adicionar_tarefa= None
    window_admin_visualizar_tarefas = None
    window_admin_visualizar_relatorios = None
    window_func_visualizar_tarefas = None
    window_func_adicionar_relatorio = None
    window_func_visualizar_relatorios = None

    while True:
        window,event,values = sg.read_all_windows()
```

Cliente: app

```
25
26          if window == window_login and event == sg.WIN_CLOSED:
27              break
28          if window == window_login and event== "Continuar":
29
30              username = window_login["username"].get()
31              password = window_login["password"].get()
32              statusLogin= login(username,password)
33
34              if statusLogin.get("status") == True:
35                  window_login.hide()
36                  if statusLogin.get("tipo") == "admin":
37                      window_menu_admin = windows.menu_admin_window()
38                  else:
39                      window_menu_func = windows.menu_func_window()
40              else:
41                  window_login["incorrect_text"].update(visible=True)
42          if (window==window_menu_admin or window==window_menu_func) and (event==sg.WIN_CLOSED or event=="Sair")
    :
43              window.close()
44              window_menu_admin = None
45              window_menu_func = None
46              window_login.un_hide()
```

Cliente: app

```
47              window_login["incorrect_text"].update(visible=False)
48              window_login["username"].SetFocus()
49              window_login["username"].update("")
50              window_login["password"].Update("")
51          if event == sg.WIN_CLOSED or event== "Voltar":
52              if window_menu_admin != None:
53                  window_menu_admin.un_hide()
54              elif window_menu_func != None:
55                  window_menu_func.un_hide()
56              window.close()
57              window_admin_adicionar_func = None
58              window_admin_visualizar_func = None
59              window_admin_adicionar_tarefa= None
60              window_admin_visualizar_tarefas = None
61              window_admin_visualizar_relatorios = None
62              window_func_visualizar_tarefas = None
63              window_func_adicionar_relatorio = None
64              window_func_visualizar_relatorios = None
65
66          if window == window_menu_admin and event == "Adicionar Funcionario":
67              window_menu_admin.hide()
68              window_admin_adicionar_func = windows.adicionar_func_window()
69
```

Cliente: app

```
70          if window == window_menu_admin and event == "Visualizar Funcionarios":
71              window_menu_admin.hide()
72              window_admin_visualizar_func = windows.visualizar_funcs_window()
73
74          if window == window_menu_admin and event == "Criar Tarefa":
75              window_menu_admin.hide()
76              window_admin_adicionar_tarefa,numFuncionarios,listaFuncionarios = windows.adicionar_tarefa_window
        ()
77
78          if window == window_menu_admin and event == "Visualizar Tarefas":
79              window_menu_admin.hide()
80              window_admin_visualizar_tarefas = windows.visualizar_tarefas_window()
81
82          if window == window_menu_admin and event == "Visualizar Relatorios":
83              window_menu_admin.hide()
84              window_admin_visualizar_relatorios = windows.visualizar_relatorios_window()
85
86          if window == window_menu_func and event == "Visualizar Tarefas":
87              window_menu_func.hide()
88              window_func_visualizar_tarefas = windows.visualizar_tarefas_funcionario_window()
89
90          if window == window_menu_func and event == "Adicionar Relatorio":
91              window_menu_func.hide()
```

UNEB
IVERSIDADE DO ESTADO DA BAHIA

Cliente: app

```
92              window_func_adicionar_relatorio,numTarefas, listasTarefas = windows.adicionar_relatorio_window()
93
94          if window == window_menu_func and event == "Visualizar Relatorios":
95              window_menu_func.hide()
96              window_func_visualizar_relatorios = windows.visualizar_relatorios_funcionario_window()
97
98          if event == "Cadastrar":
99              status=None
100             if  window== window_admin_adicionar_func:
101                 nomeFunc=values["nomeFunc"]
102                 senhaFunc=values["senhaFunc"]
103                 status = admin_view.adicionarFuncionario(nomeFunc,senhaFunc)
104             elif window == window_admin_adicionar_tarefa:
105                 nomeTarefa = values["nomeTarefa"]
106                 descricao = values["descricao"]
107                 data_prevista_conclusao = values["data_prevista_conclusao"]
108                 funcionariosId = []
109                 i = 0
110                 while i < numFuncionarios:
111                     if values[f"{i}"]:
112                         funcionariosId.append(listaFuncionarios[i].get("id"))
113                     i+=1
114                 status=admin_view.criarTarefa(nomeTarefa,descricao,data_prevista_conclusao,funcionariosId)
```

Cliente: app

```
115             elif window==window_func_adicionar_relatorio:
116                 textoRelatorio=values["texto_relatorio"]
117                 idTarefa=[]
118                 i=0
119                 while i<numTarefas:
120                     if values[f"{i}"]:
121                         idTarefa.append(listasTarefas[i].get("id"))
122                     i+=1
123                 status=funcionario_view.adicionarRelatorio(idTarefa[0],textoRelatorio)
124             if status:
125                 window["status_true"].update(visible=True)
126             else:
127                 window["status_false"].update(visible=True)
128     window_login.close()
129
130 def login(username,password):
131
132     request = requests.post(url+"/auth", json={"username": username, "password": password})
133
134     if request.status_code == 200:
135         # Abre o arquivo user.json para escrita
136         file = open("user.json", "w")
137         # Escreve os dados da resposta do servidor no arquivo user.json
```

Cliente: app

```
138         json.dump(request.json(), file)
139         # Fecha o arquivo
140         file.close()
141         if request.json()["tipo"] == "admin":
142             return {
143                 "status": True,
144                 "tipo":"admin"
145             }
146         elif request.json()["tipo"] == "funcionario":
147             return{
148                 "status": True,
149                 "tipo":"funcionario"
150             }
151     else:
152         return{
153             "status": False
154         }
155
156 if __name__ == "__main__":
157     main()
```

Cliente: windows

```python
import PySimpleGUI as sg
from view import admin_view
from view import funcionario_view

def login_window():
    sg.theme('GreenTan')
    layout = [
        [sg.Text("Nome de usuario")],
        [sg.Input(key="username")],
        [sg.Text("Senha")],
        [sg.Input(password_char="*",key="password")],
        [sg.Button("Continuar")],
        [sg.Text("Usuario ou Senha Incorretos!", text_color="red",visible=False, key="incorrect_text")]
    ]

    return sg.Window("Login", layout=layout, finalize=True)

def menu_admin_window():
    sg.theme('GreenTan')
    layout = [
        [sg.Button("Adicionar Funcionario")],
        [sg.Button("Visualizar Funcionarios")],
        [sg.Button("Criar Tarefa")],
        [sg.Button("Visualizar Tarefas")],
```

UNEB
IVERSIDADE DO
TADO DA BAHIA

Cliente: windows

```
25          [sg.Button("Visualizar Relatorios")],
26          [sg.Button("Sair")]
27      ]
28
29      return sg.Window("Menu Admin", layout=layout, finalize=True)
30
31  def menu_func_window():
32      sg.theme('GreenTan')
33      layout = [
34          [sg.Button("Visualizar Tarefas")],
35          [sg.Button("Adicionar Relatorio")],
36          [sg.Button("Visualizar Relatorios")],
37          [sg.Button("Sair")]
38      ]
39
40      return sg.Window("Menu Funcionario", layout=layout, finalize=True)
41
42  def adicionar_func_window():
43      sg.theme('GreenTan')
44      layout = [
45          [sg.Text("Nome do funcionario")],
46          [sg.Input(key="nomeFunc")],
47          [sg.Text("Senha do funcionario")],
```

Cliente: windows

```
48          [sg.Input(key="senhaFunc")],
49          [sg.Text("Relatorio cadastrado com sucesso!", text_color="white", background_color="green", visible=
        False, key="status_true")],
50          [sg.Text("Erro ao cadastrar o relatorio! \nTente Novamente!", text_color="white", background_color="
        red", visible=False, key="status_false")],
51          [sg.Button("Cadastrar")],
52          [sg.Button("Voltar")]
53      ]
54
55      return sg.Window("Adicionar Funcionario", layout=layout, finalize=True)
56
57  def adicionar_tarefa_window():
58      sg.theme('GreenTan')
59      layout = [
60          [sg.Text("Nome da Tarefa")],
61          [sg.Input(key="nomeTarefa")],
62          [sg.Text("Descricao da tarefa")],
63          [sg.Input(key="descricao")],
64          [sg.Text("Data Prevista de Conclusao (dd-mm-aa)")],
65          [sg.Input(key="data_prevista_conclusao")],
66          [sg.Text("Selecione os Funcionarios:")]
67      ]
68
```

Cliente: windows

```
69      listaFuncionarios = admin_view.visualizarFuncionarios()
70      c = 0
71      column = []
72
73      if listaFuncionarios == []:
74          column.append([sg.Text("Sem funcionarios cadastrados!")])
75      else:
76          for funcionario in listaFuncionarios:
77              column.append([sg.Checkbox(funcionario["nome"],key="{}".format(c))])
78              c += 1
79
80      layout.append([sg.Column(column, size=(200, 200), scrollable=True, vertical_scroll_only=True)])
81
82      layout.append([sg.Text("Tarefa cadastrado com sucesso!", text_color="white", background_color="green",
          visible=False, key="status_true")])
83
84      layout.append([sg.Text("Erro ao cadastrar a tarefa! \nTente Novamente!", text_color="white",
          background_color="red", visible=False, key="status_false")])
85
86      layout.append([sg.Button("Cadastrar")])
87
88      layout.append([sg.Button("Voltar")])
89
```

Cliente: windows

```
90      return sg.Window("Criar Tarefa", layout=layout, finalize=True), c, listaFuncionarios
91
92  def visualizar_funcs_window():
93      sg.theme('GreenTan')
94      layout = [
95          [sg.Text("Funcionarios: ")],
96      ]
97
98      listaFuncionarios = admin_view.visualizarFuncionarios()
99
100     if listaFuncionarios == []:
101         layout.append([sg.Text("Sem funcionarios cadastrados!")])
102     else:
103         column = []
104         for funcionario in listaFuncionarios:
105             column.append([sg.Text("-----------------------------")])
106             for key, value in funcionario.items():
107                 column.append([sg.Text("{}: {}".format(key, value))])
108         layout.append([sg.Column(column, size=(500, 500), scrollable=True, vertical_scroll_only=True)])
109
110     layout.append([sg.Button("Voltar")])
111
112     return sg.Window("Visualizar Funcionarios", layout=layout, finalize=True)
```

Cliente: windows

```
113
114 def visualizar_tarefas_window():
115     sg.theme('GreenTan')
116     layout = [
117         [sg.Text("Tarefas: ")],
118     ]
119
120     listaTarefas = admin_view.visualizarTarefas()
121
122     if listaTarefas == []:
123         layout.append([sg.Text("Sem tarefas cadastradas!")])
124     else:
125         column = []
126         for tarefa in listaTarefas:
127             column.append([sg.Text("-----------------------------")])
128             for key, value in tarefa.items():
129                 if key == "funcionarios":
130                     column.append([sg.Text("Funcionarios:")])
131                     for funcionario in value:
132                         column.append([sg.Text(funcionario)])
133                 else:
134                     column.append([sg.Text("{}: {}".format(key, value))])
135         layout.append([sg.Column(column, size=(500, 500), scrollable=True, vertical_scroll_only=True)])
```

Cliente: windows

```
136
137     layout.append([sg.Button("Voltar")])
138
139     return sg.Window("Visualizar Tarefas", layout=layout, finalize=True)
140
141 def visualizar_relatorios_window():
142     sg.theme('GreenTan')
143     layout = [
144         [sg.Text("Relatorios: ")],
145     ]
146
147     listaRelatorios = admin_view.visualizarRelatorios()
148
149     if listaRelatorios == []:
150         layout.append([sg.Text("Sem relatorios cadastrados!")])
151     else:
152         column = []
153         for relatorio in listaRelatorios:
154             column.append([sg.Text("-----------------------------")])
155             for key, value in relatorio.items():
156                 column.append([sg.Text("{}: {}".format(key, value))])
157         layout.append([sg.Column(column, size=(500, 500),scrollable=True, vertical_scroll_only=True)])
158
```

Cliente: windows

```
159     layout.append([sg.Button("Voltar")])
160
161     return sg.Window("Visualizar Relatorios", layout=layout, finalize=True)
162
163 def visualizar_tarefas_funcionario_window():
164     sg.theme('GreenTan')
165     layout = [
166         [sg.Text("Tarefas: ")],
167     ]
168
169     listaTarefas = funcionario_view.visualizarTarefas()
170
171     if listaTarefas == []:
172         layout.append([sg.Text("Sem tarefas cadastrados!")])
173     else:
174         column = []
175         for tarefa in listaTarefas:
176             column.append([sg.Text("-----------------------------")])
177             for key, value in tarefa.items():
178                 column.append([sg.Text("{}: {}".format(key, value))])
179         layout.append([sg.Column(column, size=(500, 500),scrollable=True, vertical_scroll_only=True)])
180
181     layout.append([sg.Button("Voltar")])
```

Cliente: windows

```
182
183     return sg.Window("Visualizar Tarefas", layout=layout, finalize=True)
184
185 def adicionar_relatorio_window():
186     sg.theme('GreenTan')
187     layout = [[sg.Text("Selecione a tarefa do relatorio")]]
188
189     listaTarefas = funcionario_view.visualizarTarefas()
190     c = 0
191     column = []
192
193     if listaTarefas == []:
194         column.append([sg.Text("Sem tarefas cadastrados!")])
195     else:
196         for tarefa in listaTarefas:
197             column.append([sg.Radio(tarefa["nome"], group_id=1, key="{}".format(c))])
198             c += 1
199
200     layout.append([sg.Column(column, size=(400, 200), scrollable=True, vertical_scroll_only=True)])
201
202     layout.append([sg.Text("Texto do relatorio")])
203     layout.append([sg.Input(key="texto_relatorio")])
204
```

Cliente: windows

```
205     layout.append([sg.Text("Relatorio cadastrado com sucesso!", text_color="white", background_color="green",
          visible=False, key="status_true")])
206
207     layout.append([sg.Text("Erro ao cadastrar relatorio! \nTente Novamente!", text_color="white",
          background_color="red", visible=False, key="status_false")])
208
209     layout.append([sg.Button("Cadastrar")])
210
211     layout.append([sg.Button("Voltar")])
212
213     return sg.Window("Adicionar relatorio", layout=layout, finalize=True), c, listaTarefas
214
215 def visualizar_relatorios_funcionario_window():
216     sg.theme('GreenTan')
217     layout = [
218         [sg.Text("Relatorios: ")],
219     ]
220
221     listaRelatorios = funcionario_view.visualizarRelatorios()
222
223     if listaRelatorios == []:
224         layout.append([sg.Text("Sem relatorios cadastrados!")])
225     else:
```

Cliente: windows

```
226         column = []
227         for relatorio in listaRelatorios:
228             column.append([sg.Text("-----------------------------")])
229             for key, value in relatorio.items():
230                 column.append([sg.Text("{}: {}".format(key, value))])
231         layout.append([sg.Column(column, size=(500, 500),scrollable=True, vertical_scroll_only=True)])
232 
233     layout.append([sg.Button("Voltar")])
234 
235     return sg.Window("Visualizar Relatorios", layout=layout, finalize=True)
236 
```

Cliente: admin_view

```python
import requests
import json

url = "http://127.0.0.1:8000"

def adicionarFuncionario(usernameFunc,passwordFunc):
    with open("user.json", "r") as file:
        user_dados = json.load(file)
        token = user_dados["token"]

    request = requests.put(url+"/admin/funcionarios?token="+token, json={"username": usernameFunc, "password":
       passwordFunc})

    if request.status_code == 200:
       return True
    else:
       return False


def visualizarFuncionarios():

    with open("user.json", "r") as file:
        user_dados = json.load(file)
        token = user_dados["token"]
```

Cliente: admin_view

```
24
25      request = requests.get(url+"/admin/funcionarios?token="+token)
26      return request.json()
27
28  def criarTarefa(nome,descricao,data_prevista_conclusao,funcionariosId):
29
30      with open("user.json", "r") as file:
31          user_dados = json.load(file)
32          token = user_dados["token"]
33
34      tarefa = {
35          "nome":nome,
36          "descricao": descricao,
37          "data_prevista_conclusao": data_prevista_conclusao,
38          "funcionariosId": funcionariosId
39      }
40      request = requests.put(url+"/admin/tarefas?token="+token, json=tarefa)
41
42      if request.status_code == 200:
43          return True
44      else:
45          return False
46
```

Cliente: admin_view

```
47 def visualizarTarefas():
48     with open("user.json", "r") as file:
49         user_dados = json.load(file)
50         token = user_dados["token"]
51 
52     request = requests.get(url+"/admin/tarefas?token="+token)
53 
54     return request.json()
55 def visualizarRelatorios():
56     with open("user.json", "r") as file:
57         user_dados = json.load(file)
58         token = user_dados["token"]
59 
60     request = requests.get(url+"/admin/relatorios?token="+token)
61 
62     return request.json()
```

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Cliente: funcionario_view

```python
import requests
import json
from datetime import date

url = "http://127.0.0.1:8000"

def visualizarTarefas():

    with open("user.json", "r") as file:
        user_dados = json.load(file)
        token = user_dados["token"]

    request = requests.get(f"{url}/funcionario/tarefas?token={token}")
    return request.json()
def adicionarRelatorio(idTarefa, textoRelatorio):
    with open("user.json", "r") as file:
        user_dados = json.load(file)
        idUser = user_dados["id"]
        token = user_dados["token"]

    request = requests.get(f"{url}/funcionario/tarefas?token={token}")

    dataAtual = date.today().strftime("%d-%m-%Y")

```

Cliente: funcionario_view

```
25      relatorio = {
26          "id_user": idUser,
27          "id_tarefa": int(idTarefa),
28          "texto": textoRelatorio,
29          "data_criacao": dataAtual
30      }
31
32      request = requests.put(f"{url}/funcionario/relatorios?token={token}", json=relatorio)
33
34      if request.status_code == 200:
35          return True
36      else:
37          return False
38
39  def visualizarRelatorios():
40      with open("user.json", "r") as file:
41          user_dados = json.load(file)
42          token = user_dados["token"]
43
44      request = requests.get(url+"/funcionario/relatorios?token="+token)
45
46      return request.json()
```

Conclusões Finais

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Conclusões Finais

Depois dos avancos obtidos e as metas estabelecidas no crédito anterior, criamos na parte do servidor à adicao de um token de validacao para a seguranca do sistema,e na parte do cliente a criacao de uma interface grafica, que permitiu a visualizacao do programa.

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Referências

Referências

- 9. classes — documentação python 3.12.3. https://docs.python.org/pt-br/3/tutorial/classes.html#classes, 2024. Accessed: 2024-06-02.
- Como criar apis em python usando fastapi | alura. https://www.alura.com.br/artigos/como-criar-apis-python-usando-fastapi, 2024. Accessed: 2024-06-01.

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Referências

- First steps - fastapi. https://fastapi.tiangolo.com/tutorial/first-steps/, 2024. Accessed: 2024-06-01.
- Curso de banco de dados mysql - youtube. https://www.youtube.com/playlist?list=PLHz_AreHm4dkBs-795Dsgvau_ekxg8g1r, 2024. Accessed: 2024-06-01.

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Referências

- Mysql.
  https://pythoniluminado.netlify.app/mysql#evitando-sql-injection, 2024.
  Accessed: 2024-06-01.
- Gerenciando banco de dados sqlite3 com python - parte 1 por regis da silva#pythonclub.
  https://pythonclub.com.br/gerenciando-banco-dados-sqlite3-python-parte1.html, 2024.
  Accessed: 2024-06-01.

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Referências

- Projetos com python orientado a objetos — supygirls 2.0.0 documentation.
  https://supygirls.readthedocs.io/en/latest/intro_comp/PythonOO.html, 2024.
  Accessed: 2024-06-01.
- O que é fastapi.
  https://www.treinaweb.com.br/blog/o-que-e-fastapi.
  Accessed: 2024-06-02.

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Referências

- Fastapi.
  https://fastapi.tiangolo.com/, 2024.
  Accessed: 2024-06-02.
- Real Python.
  Biblioteca de solicitações do python (guia) – python real.
  https://realpython.com/python-requests/, 2024.
  Accessed: 2024-06-02.

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Referências

Hugo Habbema.
Pydantic. simplificando a validação de dados em python.
https://medium.com/@habbema/pydantic-23fe91b5749b, 2024.
Accessed: 2024-06-02.

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA