Код проекта IOT

## Файл: print\_code.py

import os

import logging

from docx import Document

# Настройка логирования

logging.basicConfig(

level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s'

)

logger = logging.getLogger(\_\_name\_\_)

def add\_file\_content\_to\_document(

doc, directory, extensions, exclude\_dirs=None

):

if exclude\_dirs is None:

exclude\_dirs = []

for root, dirs, files in os.walk(directory):

# Исключаем заданные директории из обхода

dirs[:] = [d for d in dirs if d not in exclude\_dirs]

for file in files:

if file.endswith(tuple(extensions)):

file\_path = os.path.join(root, file)

logger.info(f'Добавление файла: {file\_path}')

# Добавление названия директории и файла

doc.add\_heading(

'Файл: ' + file\_path.replace(directory, ''), level=2

)

with open(file\_path, 'r', encoding='utf-8') as f:

# Добавление содержимого файла

content = f.read()

paragraphs = content.split('\n')

for paragraph in paragraphs:

doc.add\_paragraph(paragraph)

doc.add\_page\_break() # Добавляем разрыв страницы после каждого файла # noqa

def create\_word\_document(

directory, extensions,

filename="IOTProjectCode.docx", exclude\_dirs=None

):

logger.info('Создание документа Word...')

doc = Document()

doc.add\_heading('Код проекта IOT', 0)

add\_file\_content\_to\_document(doc, directory, extensions, exclude\_dirs)

# Сохранение документа

save\_path = f"C:/dev/kiosk/{filename}"

doc.save(save\_path)

logger.info(f"Документ '{filename}' успешно сохранён в {save\_path}.")

# Пример использования

if \_\_name\_\_ == "\_\_main\_\_":

directory = "C:\\dev\\kiosk\\"

extensions = ['.py', '.html'] # Расширения файлов для включения # noqa

exclude\_dirs = ['venv', 'migrations'] # Список каталогов для исключения

create\_word\_document(

directory, extensions, "IOTProjectCode.docx", exclude\_dirs

)

## Файл: app\database.py

# app/database.py

from sqlalchemy import create\_engine

DATABASE\_URL = "postgresql://user:password@localhost/dbname"

engine = create\_engine(DATABASE\_URL)

## Файл: app\dependencies.py

# app/dependencies

from sqlalchemy.orm import sessionmaker

from .database import engine

def get\_db():

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

db = SessionLocal()

try:

yield db

finally:

db.close()

## Файл: app\main.py

# app/main.py

from datetime import datetime

from fastapi import FastAPI, Depends, HTTPException, Form, Request

from fastapi.responses import HTMLResponse

from fastapi.staticfiles import StaticFiles

from fastapi.templating import Jinja2Templates

from sqlalchemy.orm import Session

from .dependencies import get\_db

from .database import engine

from .models import Base, UserEquipment

app = FastAPI()

# Создание таблиц

Base.metadata.create\_all(engine)

app.mount("/static", StaticFiles(directory="static"), name="static")

templates = Jinja2Templates(directory="templates")

@app.post("/login")

async def login(

username: str = Form(...),

password: str = Form(...),

db: Session = Depends(get\_db)

):

query = "SELECT \* FROM users WHERE username = :username AND password = :password" # noqa

result = db.execute(

query, {'username': username, 'password': password}

).fetchone()

if result:

return {"user\_id": result.user\_id}

else:

raise HTTPException(status\_code=400, detail="Invalid credentials")

@app.get("/equipment/")

def list\_equipment(db: Session = Depends(get\_db)):

result = db.execute("SELECT \* FROM equipment")

return [dict(row) for row in result]

@app.get("/login/", response\_class=HTMLResponse)

def get\_login(request: Request):

return templates.TemplateResponse("login.html", {"request": request})

@app.get("/dashboard/", response\_class=HTMLResponse)

def get\_dashboard(request: Request):

return templates.TemplateResponse("dashboard.html", {"request": request})

@app.post("/start\_shift/")

async def start\_shift(user\_id: int, db: Session = Depends(get\_db)):

current\_time = datetime.now()

db.execute(

"INSERT INTO user\_sessions (user\_id, check\_in\_time) VALUES (:user\_id, :time)", # noqa

{'user\_id': user\_id, 'time': current\_time}

)

db.commit()

return {"message": "Shift started", "time": current\_time}

@app.post("/end\_shift/")

async def end\_shift(user\_id: int, db: Session = Depends(get\_db)):

current\_time = datetime.now()

db.execute(

"UPDATE user\_sessions SET check\_out\_time = :time WHERE user\_id = :user\_id", # noqa

{'time': current\_time, 'user\_id': user\_id}

)

db.commit()

return {"message": "Shift ended", "time": current\_time}

@app.post("/assign\_downtime/")

async def assign\_downtime(

user\_id: int, downtime\_id: int, type\_id: int, db: Session = Depends(get\_db)

):

db.execute(

"UPDATE workflow SET answer\_id = :type\_id WHERE id = :downtime\_id AND equipment\_id IN (SELECT equipment\_id FROM user\_equipment WHERE user\_id = :user\_id)", # noqa

{'type\_id': type\_id, 'downtime\_id': downtime\_id, 'user\_id': user\_id}

)

db.commit()

return {

"message": "Downtime assigned",

"downtime\_id": downtime\_id,

"type\_id": type\_id

}

@app.post("/assign\_equipment/")

async def assign\_equipment(

user\_id: int, equipment\_id: int, db: Session = Depends(get\_db)

):

new\_assignment = UserEquipment(

user\_id=user\_id, equipment\_id=equipment\_id, start\_time=datetime.now()

)

db.add(new\_assignment)

db.commit()

return {

"message": "Equipment assigned successfully",

"user\_id": user\_id,

"equipment\_id": equipment\_id

}

@app.post("/release\_equipment/")

async def release\_equipment(

user\_id: int, equipment\_id: int, db: Session = Depends(get\_db)

):

assignment = db.query(UserEquipment).filter(

UserEquipment.user\_id == user\_id,

UserEquipment.equipment\_id == equipment\_id

).first()

if assignment:

assignment.end\_time = datetime.now()

db.commit()

return {

"message": "Equipment released",

"user\_id": user\_id,

"equipment\_id": equipment\_id

}

else:

raise HTTPException(status\_code=404, detail="Assignment not found")

## Файл: app\models.py

# app/models.py

from sqlalchemy import Column, Integer, ForeignKey, DateTime

from sqlalchemy.orm import declarative\_base # , relationship

Base = declarative\_base()

class UserSession(Base):

\_\_tablename\_\_ = 'user\_sessions'

id = Column(Integer, primary\_key=True)

user\_id = Column(Integer, ForeignKey('users.id'))

check\_in\_time = Column(DateTime)

check\_out\_time = Column(DateTime)

class UserDowntime(Base):

\_\_tablename\_\_ = 'user\_downtimes'

id = Column(Integer, primary\_key=True)

session\_id = Column(Integer, ForeignKey('user\_sessions.id'))

workflow\_id = Column(Integer, ForeignKey('workflow.id'))

type\_id = Column(Integer, ForeignKey('answers\_list.id'))

class UserEquipment(Base):

\_\_tablename\_\_ = 'user\_equipment'

id = Column(Integer, primary\_key=True)

user\_id = Column(Integer, ForeignKey('users.id'))

equipment\_id = Column(Integer, ForeignKey('equipment.id'))

start\_time = Column(DateTime)

end\_time = Column(DateTime)

## Файл: app\\_\_init\_\_.py

# app/\_\_init\_\_.py

## Файл: templates\dashboard.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Панель управления</title>

<link rel="stylesheet" href="/static/css/style.css">

</head>

<body>

<div class="dashboard-container">

<h1>Панель управления</h1>

<button onclick="startShift()">Начать смену</button>

<button onclick="endShift()">Завершить смену</button>

<h2>Выберите простой</h2>

<select id="downtimeSelect"></select>

<select id="downtimeTypeSelect"></select>

<button onclick="assignDowntime()">Применить простой</button>

<h2>Управление оборудованием</h2>

<select id="equipmentSelect"></select>

<button onclick="assignEquipment()">Назначить оборудование</button>

<button onclick="releaseEquipment()">Освободить оборудование</button>

</div>

<script>

// Загрузка списка простоев и оборудования при загрузке страницы

function loadDowntimes() {

fetch('/downtimes')

.then(response => response.json())

.then(data => {

const downtimeSelect = document.getElementById('downtimeSelect');

data.forEach(downtime => {

const option = document.createElement('option');

option.value = downtime.id;

option.textContent = downtime.name;

downtimeSelect.appendChild(option);

});

})

.catch(error => console.error('Ошибка:', error));

fetch('/downtime\_types')

.then(response => response.json())

.then(types => {

const typeSelect = document.getElementById('downtimeTypeSelect');

types.forEach(type => {

const option = document.createElement('option');

option.value = type.id;

option.textContent = type.description;

typeSelect.appendChild(option);

});

})

.catch(error => console.error('Ошибка при загрузке типов:', error));

fetch('/equipment/')

.then(response => response.json())

.then(data => {

const equipmentSelect = document.getElementById('equipmentSelect');

data.forEach(equipment => {

const option = document.createElement('option');

option.value = equipment.id;

option.textContent = equipment.name;

equipmentSelect.appendChild(option);

});

})

.catch(error => console.error('Ошибка при загрузке оборудования:', error));

}

function startShift() {

const userId = sessionStorage.getItem('userId');

fetch(`/start\_shift/`, {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ user\_id: userId })

})

.then(response => response.json())

.then(data => alert('Смена начата: ' + data.time))

.catch(error => console.error('Ошибка:', error));

}

function endShift() {

const userId = sessionStorage.getItem('userId');

fetch(`/end\_shift/`, {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ user\_id: userId })

})

.then(response => response.json())

.then(data => alert('Смена завершена: ' + data.time))

.catch(error => console.error('Ошибка:', error));

}

function assignDowntime() {

const userId = sessionStorage.getItem('userId');

const downtimeId = document.getElementById('downtimeSelect').value;

const typeId = document.getElementById('downtimeTypeSelect').value;

fetch('/assign\_downtime/', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ user\_id: userId, downtime\_id: downtimeId, type\_id: typeId })

})

.then(response => response.json())

.then(data => alert('Простой применен: ' + data.message))

.catch(error => console.error('Ошибка:', error));

}

function assignEquipment() {

const userId = sessionStorage.getItem('userId');

const equipmentId = document.getElementById('equipmentSelect').value;

fetch('/assign\_equipment/', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ user\_id: userId, equipment\_id: equipmentId })

})

.then(response => response.json())

.then(data => alert('Оборудование назначено: ' + data.message))

.catch(error => console.error('Ошибка:', error));

}

function releaseEquipment() {

const userId = sessionStorage.getItem('userId');

const equipmentId = document.getElementById('equipmentSelect').value;

fetch('/release\_equipment/', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({ user\_id: userId, equipment\_id: equipmentId })

})

.then(response => response.json())

.then(data => alert('Оборудование освобождено: ' + data.message))

.catch(error => console.error('Ошибка:', error));

}

window.onload = loadDowntimes;

</script>

</body>

</html>

## Файл: templates\login.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Вход в систему</title>

<link rel="stylesheet" href="/static/css/style.css">

</head>

<body>

<div class="login-container">

<h2>Вход в систему</h2>

<form id="loginForm">

<div class="form-group">

<label for="username">Имя пользователя:</label>

<input type="text" id="username" name="username" required>

</div>

<div class="form-group">

<label for="password">Пароль:</label>

<input type="password" id="password" name="password" required>

</div>

<div class="form-group">

<button type="submit">Вход</button>

</div>

</form>

</div>

<script>

document.getElementById('loginForm').addEventListener('submit', function(event) {

event.preventDefault();

const username = document.getElementById('username').value;

const password = document.getElementById('password').value;

fetch('/login', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ username, password })

})

.then(response => response.json())

.then(data => {

if (data.user\_id) {

sessionStorage.setItem('userId', data.user\_id);

window.location.href = '/dashboard.html'; // Перенаправление на главную страницу

} else {

alert('Ошибка входа');

}

})

.catch(error => console.error('Ошибка:', error));

});

</script>

</body>

</html>