Relatório do Projeto de Sistema de Gerenciamento de Dados de Ações

Projeto Realizado para a disciplina de: BDNoSQL 1º Ano do Curso de DS

Projeto Realizado por: Dmytro Bohutskyy

Índice:

Resumo do Projeto

Este projeto implementa um sistema de gestão de dados de ações com autenticação de utilizadores utilizando Flask e MongoDB (via PyMongo). O sistema diferencia entre utilizadores comuns e administradores, oferecendo páginas e funcionalidades específicas para cada tipo de utilizador.

Descrição do Projeto

Tecnologias Utilizadas

- **Flask:** Micro framework para Python utilizado para criar as rotas e gerir as requisições HTTP.
- **PyMongo:** Biblioteca para interagir com o MongoDB a partir do Python.
- **yFinance:** Biblioteca para obter dados financeiros históricos.
- **Bokeh:** Biblioteca para criar gráficos interativos para visualização de dados.
- **Pandas:** Biblioteca para manipulação e análise de dados.
- **Requests:** Biblioteca para fazer requisições HTTP em Python, utilizada para integrar com a API da MarketAux.
- **HTML/CSS:** Para a criação das interfaces web.
- **JavaScript:** Para interatividade e manipulação do DOM.
- **SweetAlert2:** Biblioteca para exibir alertas e mensagens de forma estilizada.
- **Marketaux API:** API utilizada para obter notícias relacionadas às ações.

Estrutura do Sistema

Base de Dados MongoDB

O MongoDB é utilizado para armazenar informações dos utilizadores, dados históricos de ações, alertas, recomendações, preferências dos utilizadores, logs do sistema e notícias. Cada uma dessas informações é armazenada em coleções separadas, permitindo uma gestão eficiente e organizada dos dados.

Flask Application

A aplicação Flask gere as rotas, a lógica de autenticação e a comunicação com a base de dados. A estrutura da aplicação está organizada para garantir uma fácil manutenção e escalabilidade.

Autenticação: Utiliza sessões para gerir o estado do utilizador e garantir que apenas utilizadores autenticados podem aceder a determinadas funcionalidades.
Rotas: Definidas para gerir as várias funcionalidades da aplicação, desde o login até à visualização de dados de ações e gestão de alertas.

Funcionalidades Implementadas

1. Página de Login: Formulário para que os utilizadores introduzam as suas credenciais (nome de utilizador e senha).
2. Autenticação: Verificação de credenciais usando MongoDB e redirecionamento com base na função do utilizador.
3. Página de Administração: Disponível apenas para administradores, permitindo a visualização e gestão de dados.
4. Busca e Armazenamento de Dados de Ações: Busca dados históricos de ações utilizando a biblioteca yFinance e armazena-os no MongoDB.
5. Recomendações de Compra/Venda: Baseado nas médias móveis dos preços das ações.
6. Visualização de Dados: Criação de gráficos interativos utilizando Bokeh para a visualização dos dados históricos de ações.
7. Integração de Notícias: Busca e armazena notícias relacionadas às ações utilizando a API marketaux.
8. Preferências do Utilizador: Armazenamento de preferências como tema e idioma.

Descrição das Rotas

- Rota /: Página inicial que redireciona para a página de login se o utilizador não estiver autenticado.
- Rota /login: Gere o login dos utilizadores, verificando as credenciais no MongoDB.
- Rota /logout: Implementa a funcionalidade de logout.
- Rota /fetch: Permite a busca e armazenamento de dados de ações.
- Rota /show: Exibe dados históricos de ações e recomendações baseadas nesses dados.
- Rota /alerts: Gere alertas configurados pelos utilizadores.
- Rota /logs: Visualiza logs de atividades do sistema.
- Rota /change_theme/<theme>: Permite ao utilizador mudar o tema da aplicação.
- Rota /change_language/<lang>: Permite ao utilizador mudar o idioma da aplicação

Base de Dados

Modelo de Entidade-Relacionamento (ER)

Entidade Users:

- id (Primary Key): Identificador único.
- username: String, único.
- password: String.
- email: String.
- created_at: Date.
- updated_at: Date.
- is_admin: Boolean.
- preferences: Objeto com tema e tipo de gráfico padrão.

Entidade Alerts:

- user_id: String.
- symbol: String.
- condition: String.
- threshold: Float.
- created_at: Date.
- active: Boolean.

Entidade Recommendations:

- symbol: String.
- recommendation: String.
- explanation: String.
- date: Date.

Entidade Stocks:

- symbol: String, único.
- name: String.
- sector: String.
- exchange: String.
- history: Lista de dados históricos (contendo data, preço de abertura, preço alto, preço baixo, preço de fechamento, volume).

Entidade System Logs:

- timestamp: Date.
- message: String.
- severity: String.
- user_id: String.

Entidade User Preferences:

- user_id: String.
- theme: String.
- language: String.
- default_chart_type: String.

Entidade News:

- symbol: String.
- title: String.
- description: String.
- url: String.
- publishedAt: Date.

Entidade Sentiment Analysis:

- symbol: String.
- sentiment: String.
- timestamp: Date.

Relacionamentos

Nesta aplicação, as entidades estão estruturadas de forma a armazenar dados de utilizadores, ações, preferências, alertas, recomendações, logs do sistema, notícias e análise de sentimento. Cada entidade tem campos específicos para armazenar as informações necessárias.

Script para a Criação e Inserção de Dados Iniciais

Os scripts initdb.py e insert_data.py são responsáveis por criar e inserir dados iniciais na base de dados, como a criação de dois utilizadores (um administrador e um utilizador comum) e a inserção de dados históricos de algumas ações. Estes scripts são automaticamente executados ao iniciar o main.py, então não é necessário executá-los separadamente antes de rodar o main.py.

Diferenças da Proposta Inicial

Integração de Bokeh:

- A proposta inicial incluía o uso do Bokeh para gráficos interativos em tempo real. No entanto, o Bokeh não suporta gráficos em tempo real de forma eficiente com a tecnologia de WebSockets utilizada no Flask. Isso levou à decisão de remover essa funcionalidade específica de gráficos em tempo real.

Coleções Sessions e Visualization Data:

- As coleções Sessions e Visualization Data foram inicialmente propostas, mas devido a dificuldades técnicas e a falta de necessidade prática, essas coleções foram removidas e substituídas por sentiment_analysis e news, que agregam mais valor ao projeto.

Conclusão

Este projeto demonstra a implementação de um sistema completo de autenticação de utilizadores com diferenciação de papéis, integração com uma API financeira para obtenção de dados de ações e notícias, e visualização desses dados de forma interativa. Utilizando Flask e MongoDB, o sistema oferece uma plataforma robusta e escalável para o gerenciamento de dados de ações.

Observações Finais

O projeto inicial foi ajustado para melhor atender às necessidades reais e às limitações técnicas encontradas durante o desenvolvimento. A substituição de coleções e a adaptação do uso de tecnologias como o Bokeh foram necessárias para garantir a funcionalidade e a eficiência do sistema. O resultado é uma aplicação prática e funcional, pronta para uso e expansão futura.

Bibliografia

Flask Documentation

- Documentação oficial do Flask: https://flask.palletsprojects.com/en/2.0.x/

PyMongo Documentation

- Documentação oficial do PyMongo: https://pymongo.readthedocs.io/en/stable/

yFinance Documentation

- Documentação oficial do yFinance: https://pypi.org/project/yfinance/

Bokeh Documentation

- Documentação oficial do Bokeh: https://docs.bokeh.org/en/latest/

MarketAux API Documentation

- Documentação oficial da MarketAux API: https://marketaux.com/docs/

MongoDB Documentation

- Documentação oficial do MongoDB: https://docs.mongodb.com/

SweetAlert2 Documentation

- Documentação oficial do SweetAlert2: https://sweetalert2.github.io/

W3Schools

- Recursos gerais de HTML/CSS: https://www.w3schools.com/

Stack Overflow

- Discussões e soluções para problemas de programação: https://stackoverflow.com/

Pandas Documentation

- Documentação oficial do Pandas: https://pandas.pydata.org/docs/

Requests Documentation

- Documentação oficial do Requests: https://docs.python-requests.org/en/latest/

Estas fontes foram utilizadas para o desenvolvimento do projeto, incluindo a configuração do Flask, integração com MongoDB via PyMongo, utilização da API do yFinance para obtenção de dados financeiros, e criação de gráficos interativos com Bokeh, entre outras funcionalidades.