Universidade Atlântica

Onlib

Base de Dados II

Ruben Gonçalves (202327007), Gonçalo Rodrigues (20192358), João Almeida (202327046)
01-12-2024

Índice

Introdução

Bem-vindo ao guia de instalação e execução da aplicação OnLib. Este documento foi elaborado para fornecer um conjunto completo de instruções e informações necessárias para configurar, implementar e validar a aplicação, garantindo que todos os componentes funcionem de forma integrada e eficiente.

A OnLib é uma aplicação web inovadora que combina tecnologias modernas para oferecer uma experiência de utilizador robusta e intuitiva. Desde a gestão de catálogos de livros e empréstimos até à personalização de recomendações com base em Machine Learning, a OnLib foi concebida para atender às necessidades de bibliotecas e utilizadores de forma eficaz.

Neste guia, abordaremos os pré-requisitos necessários, a configuração do ambiente, a estrutura do backend em Python com Flask, a organização dos microserviços, e a integração com o RabbitMQ. Além disso, detalharemos o desenvolvimento do frontend com React, a interface administrativa com Tkinter, e a integração com serviços externos como a autenticação OAuth 2.0 e a API do Google Books.

Também incluímos secções sobre monitorização e logs, utilizando ferramentas como Prometheus e Elastic Stack, e a implementação de recomendações personalizadas através de algoritmos de Machine Learning. Cada secção foi cuidadosamente estruturada para fornecer instruções claras e detalhadas, facilitando a instalação e execução da aplicação.

Esperamos que este guia seja um recurso valioso para a implementação da OnLib e que ajude a maximizar o seu potencial. Se tiver alguma dúvida ou necessitar de assistência adicional, estamos à disposição para ajudar.

Onlib - Guia de Instalação e Execução

Pré-requisitos

Para garantir que o ambiente está preparado para a instalação e execução da aplicação, é necessário ter as seguintes ferramentas instaladas:

- **Node.js (v18 ou superior): Plataforma para execução de código JavaScript no servidor.**
- **MySQL (com MySQL Workbench): Sistema de gestão de bases de dados relacionais.**
- **MongoDB: Base de dados NoSQL para armazenamento de dados dinâmicos.**
- **React.js: Biblioteca JavaScript para construção de interfaces de utilizador.**
- **RabbitMQ: Sistema de mensageria para comunicação entre serviços.**
- **Python (v3.10 ou superior): Linguagem de programação utilizada no backend.**
- **Prometheus: Ferramenta de monitorização e alerta.**
- **Elastic Stack (Elasticsearch, Kibana e Filebeat): Conjunto de ferramentas para pesquisa, análise e visualização de dados.**
- **Git: Sistema de controlo de versões.**

Configuração do Ambiente

Nesta secção, são fornecidas instruções para configurar o ambiente necessário para a aplicação:

- Base de Dados:
  - Criação das bases de dados MySQL necessárias.
  - Importação do script SQL para configurar as tabelas.
- Armazenamento de Dados de Interações dos Utilizadores com MongoDB:
  - Logs de Acesso: Rastreiam ações do utilizador.
  - Histórico de Leitura: Regista os livros lidos pelos utilizadores.
  - Preferências de Utilizador: Armazena géneros literários favoritos.

Armazenar Dados de Interações dos Utilizadores utilizando o MongoDB:

Na aplicação **Onlib**, utilizamos o MongoDB para armazenar informações dinâmicas relacionadas às interações dos utilizadores, como:

Exemplos de registos em Logs:

Acesso:

```
{
        "userId": "1",
        "action": "Login efetuado",
        "timestamp": "2024-12-01T10:00:00Z"
}
```

Histórico de Leitura:

```
{
  "userId": "1",
  "bookTitle": "1984",
  "author": "George Orwell",
  "completedDate": "2024-11-30"
}
```

Preferências de Utilizador:

```
{
  "userId": "1",
  "preferredGenres": ["Ficção", "Mistério"],
  "lastUpdated": "2024-12-01T10:30:00Z"
}
```

Backend em Python com Flask

Detalha a estrutura e configuração do backend desenvolvido em Python utilizando o framework Flask:

- **Estrutura do backend Flask**: Organização dos ficheiros e diretórios.

```
onlib/
├── app.py
├── config.py
├── routes/
   ├── books.py
   ├── borrow.py
   ├── recommendations.py
   ├── returns.py
   ├── sync.py
```

- **Testar a API**:
  - Iniciar o Servidor: Comando para iniciar o servidor.

    ```
    bash
    python app.py
    ```

  - Testar os Endpoints: Exemplos de testes utilizando cURL.

    Exemplos de Testes com cURL

    - **GET /books**

      ```
      Bash
      curl -X GET http://localhost:5000/books/
      ```

    - **POST /borrow**

Bash
curl -X POST -H "Content-Type: application/json" -d '{"book_id": 1, "user_id": 123}' http://localhost:5000/borrow/

- **PUT /return**
  Bash
  curl -X PUT -H "Content-Type: application/json" -d '{"book_id": 1, "user_id": 123}' http://localhost:5000/return/

- **GET /recommendations**
  Bash
  curl -X GET http://localhost:5000/recommendations/?user_id=123

- **POST /sync**
  Bash
  curl -X POST -H "Content-Type: application/json" -d '{}' http://localhost:

Estrutura dos Microserviços

Descrição dos microserviços que compõem a aplicação:

- **Catálogo de Livros**: Gere o catálogo de livros disponíveis.
  - **Função:** Gerir o catálogo de livros disponíveis.
  - **API:** /api/books
    - **GET /api/books:** Listar todos os livros.
    - **POST /api/books:** Adicionar um novo livro.
    - **PUT /api/books/:id:** Atualizar um livro.
    - **DELETE /api/books/:id:** Remover um livro.
- **Gestão de Empréstimos**: Gere os empréstimos de livros.
  - **Função:** Gerir os empréstimos de livros.
  - **API:** /api/borrow
    - **POST /api/borrow:** Realizar o empréstimo de um livro.
    - **PUT /api/borrow/:id/return:** Devolver um livro.
    - **GET /api/borrow/user/:userId:** Verificar os empréstimos de um utilizador.
- **Recomendações**: Fornece recomendações personalizadas.
  - **Função:** Fornecer recomendações personalizadas com base no histórico do utilizador.
  - **API:** /api/recommendations
    - **GET /api/recommendations/user/:userId:** Obter recomendações para um utilizador.
- **Integração Externa**: Sincroniza dados com sistemas externos.
  - **Função:** Sincronizar dados com sistemas externos.
  - **API:** /api/sync
    - **POST /api/sync:** Sincronizar dados.

Acesso ao RabbitMQ e Validação

Instruções para instalação e validação do RabbitMQ:

- **Instalação do RabbitMQ**: Passos para instalar o RabbitMQ.
    - Certifica-te de que o RabbitMQ está instalado e a correr no teu servidor. Segue as instruções de instalação no site oficial do RabbitMQ.
- **Acesso ao RabbitMQ Management**: Como aceder à interface de gestão.
    - **Abrir a Interface de Gestão do RabbitMQ**
        - A interface de gestão está normalmente disponível em http://localhost:15672.
        - Insere o endereço no teu navegador e carrega Enter.
    - **Login na Interface de Gestão**
    - As credenciais por padrão são:
        - **Username:** guest
        - **Password:** guest
    - Depois de inseri as credenciais, clica em "Login".
- **Explorar a Interface de Gestão**: Navegação pelas filas, exchanges e bindings.
    - **Queues (Filas)**
        - Clica na aba "Queues" para ver todas as filas existentes.
        - As filas são utilizadas para armazenar mensagens enviadas pelos produtores antes de serem consumidas pelos consumidores.
    - **Exchanges**
        - Clica na aba "Exchanges" para ver todas as exchanges.
        - As exchanges são responsáveis por direcionar as mensagens para as filas corretas com base em regras de routing.
    - **Bindings**
        - Clica na aba "Bindings" dentro de uma exchange para ver como as filas estão ligadas a essa exchange.
    - **Producers e Consumers**
        - Podes verificar os produtores que estão a enviar mensagens para as filas e os consumidores que estão a ler mensagens das filas.
- **Validar o Funcionamento do RabbitMQ**: Testes para garantir o funcionamento correto.
    - **Criar uma Fila**
        - Na aba "Queues", clica em "Add a new queue".
        - Introduz o nome da fila (por exemplo, test_queue) e clica em "Add queue".
    - **Enviar uma Mensagem para a Fila**
        - Na aba "Queues", clica no nome da fila que criaste (test_queue).
        - Na secção "Publish message", insere uma mensagem (por exemplo, {"message": "Hello RabbitMQ"}) e clica em "Publish message".
    - **Verificar a Mensagem na Fila**
        - Depois de publicares a mensagem, deves ver a mensagem na secção "Messages ready".
        - Clica em "Get messages" para visualizar a mensagem.

Frontend Web Avançado

O frontend da aplicação **Onlib** foi desenvolvido utilizando o **React**, proporcionando uma interface moderna, intuitiva e responsiva para os utilizadores. O sistema permite gerir interações com o catálogo de livros, empréstimos, devoluções e preferências de forma eficiente.

Detalhes sobre o desenvolvimento do frontend da aplicação utilizando React:

- **Funcionalidades Implementadas**: Navegação, pesquisa, recomendações, gestão de empréstimos e preferências.
    - **Navegação e Busca no Catálogo**:
        - Permite explorar o catálogo de livros disponíveis.
        - Inclui uma funcionalidade de **busca avançada** para filtrar livros por título, autor ou género.
    - **Recomendações Personalizadas**:
        - Exibe recomendações baseadas no histórico de leitura e nas preferências do utilizador.
    - **Gestão de Empréstimos e Devoluções**:
        - Permite realizar **empréstimos** e efetuar a **devolução de livros** diretamente através da interface.
    - **Histórico e Preferências do Utilizador**:
        - Mostra o **histórico de leitura**, incluindo livros lidos e datas de conclusão.
        - Permite ajustar **preferências pessoais**, como géneros literários favoritos.
- **Tecnologias Utilizadas**: React, Axios, React Router, Bootstrap.
    - **React**: Framework utilizado para o desenvolvimento do frontend.
    - **Axios**: Biblioteca usada para comunicação com a API do backend.
    - **React Router**: Implementado para navegação entre páginas.
    - **Bootstrap**: Para estilização e design responsivo.
- **Acesso ao Sistema**: Passos para iniciar o frontend e backend.
- **Estrutura de Componentes**: Descrição dos principais componentes.

Para utilizar o frontend, é necessário garantir que o servidor backend também está em execução. O servidor backend funciona na porta **5000** e comunica diretamente com o frontend.

Passos para aceder ao frontend:

1. Certifica-te de que o backend está a correr:
    - Na diretoria do backend, inicia o servidor:
      bash
      python app.py
    - Verifica que o backend está acessível em:
      http://localhost:5000

2. Certifica-te de que o frontend está em execução:
    - Na diretoria do frontend, inicia o servidor:
      npm start

    - Acede ao frontend no browser:
      http://localhost:3000

- **Configuração e Execução**: Instruções para configurar e executar o sistema.

  Para configurar e executar o sistema completo:

  1. Certifica-te de que tens o **Node.js** e o **Python** instalados.
  2. No backend:
     - Instala as dependências (se necessário):
       pip install -r requirements.txt
     - Inicia o servidor:
       python app.py
  3. No frontend:
     - Instala as dependências:
       npm install

     - Inicia o servidor:
       npm start

  4. No frontend:
     - Instala as dependências:
       npm install

     - Inicia o servidor:
       npm start

- **Validação**: Testes para garantir o funcionamento das funcionalidades.
    - Certifica-te de que ambos os servidores (backend na porta **5000** e frontend na porta **3000**) estão em execução.
    - Testa as seguintes funcionalidades:
        - Busca no catálogo de livros.
        - Visualização de recomendações.
        - Empréstimo e devolução de livros.
        - Consulta do histórico de leitura e ajustes nas preferências.

O frontend foi desenvolvido com componentes modulares para garantir organização e escalabilidade. Aqui estão os principais componentes:

- **Catalog**: Exibe a lista de livros com funcionalidade de busca avançada.
- **Recommendations**: Mostra recomendações personalizadas para o utilizador.
- **LoanManagement**: Permite realizar empréstimos e devoluções.
- **UserHistory**: Exibe o histórico de leitura.
- **Preferences**: Permite visualizar e ajustar as preferências do utilizador.

Interface Administrativa com Tkinter

A aplicação **Onlib** inclui uma **interface administrativa** desenvolvida com **Tkinter**, permitindo que administradores gerenciem a biblioteca de forma eficiente através de uma aplicação desktop.

Descrição da interface administrativa desenvolvida com Tkinter:

- **Funcionalidades Implementadas**: Gestão do catálogo, empréstimos, utilizadores e relatórios.

    1. **Gestão do Catálogo de Livros**:
        - **Adicionar**: Permite inserir novos livros no catálogo.
        - **Remover**: Remove livros existentes do catálogo.
        - **Editar**: Atualiza informações como título, autor ou número de cópias disponíveis.
    2. **Gestão de Empréstimos**:
        - Exibe todos os empréstimos registados no sistema.
        - Mostra o **status** de cada livro (emprestado, disponível, devolvido).
    3. **Gestão de Utilizadores**:
        - Lista todos os utilizadores registados no sistema.
        - Permite realizar ações administrativas, como ajustar permissões.
    4. **Relatórios de Atividades**:
        - Visualiza relatórios de atividades, como:
            1. Número total de empréstimos realizados.
            2. Utilizadores mais ativos.
            3. Histórico de alterações no catálogo.

- **Localização da Interface**: Diretoria onde a interface está localizada.

    - A interface administrativa encontra-se na diretoria:
      onlib-admin/

    - O ficheiro principal para abrir a interface é:
      main.py

- **Guia de Utilização**: Passos para abrir e interagir com a interface.

    1. **Pré-requisitos**:
        - Certifica-te de que tens o Python instalado na máquina.
        - Verifica que o servidor backend está em execução na porta **5000**, pois a interface administrativa comunica diretamente com ele.
    2. **Abrir a Interface Administrativa**:
        - Navega até à diretoria onlib-admin:
            1. cd onlib-admin
        - Inicia o script principal:
            1. python main.py
    3. **Interagir com a Interface**:
        - A interface gráfica será aberta, permitindo o acesso às funcionalidades:

1. **Gerir o catálogo de livros**: Usa os botões "Adicionar", "Editar" ou "Remover".
2. **Visualizar empréstimos**: Acede à aba de empréstimos para consultar detalhes.
3. **Gerir utilizadores**: Seleciona utilizadores da lista e edita permissões ou registos.
4. **Gerar relatórios**: Utiliza a opção de relatórios para obter um resumo de atividades.

- **Configuração e Execução**: Instruções para instalar dependências e executar a interface.
  - Certifica-te de que as dependências necessárias estão instaladas:

    pip install -r requirements.txt
    (se o ficheiro requirements.txt existir na diretoria).

  - Executa o script principal (main.py) para abrir a interface.
- **Validação**: Testes para garantir o funcionamento da interface.
  - Abre a aplicação através do script main.py.
  - Realiza operações básicas, como:
    - Adicionar um livro ao catálogo e verificar no backend se foi atualizado.
    - Consultar os empréstimos e confirmar os dados apresentados.
    - Gerar relatórios e verificar os valores calculados.

Interoperabilidade e Integração com Serviços Externos

Instruções para integração com serviços externos:

- **Sistema de Autenticação OAuth 2.0**: Configuração para login com Google.
  - **Objetivo:** Permitir que outras bibliotecas e sistemas se conectem de forma segura utilizando autenticação OAuth 2.0.
  - **Implementação:** Utilização do Google Auth via Google Cloud para autenticação.
  - **Passos de Configuração:**
    - **Registar a aplicação** no Google Cloud Console.
    - **Configurar credenciais OAuth 2.0** (Client ID e Client Secret).
    - **Integrar com a aplicação** para permitir login com Google.
  - **Guia de Utilização:**
  - **Aceder ao login:** Os utilizadores podem clicar no botão "Login com Google" na página de login.
  - **Autorização:** Serão redirecionados para a página de autorização do Google para conceder permissões.
- **Integração com a API do Google Books**: Importação de dados de livros.
  - **Objetivo:** Importar dados de livros diretamente da API do Google Books.
  - **Implementação:**

- Utilizar endpoints da API do Google Books para buscar e sincronizar informações de livros.
- Exemplos de pedidos:

```
Python

import requests

def fetch_books(query):
response=equests.get(f'https://www.googleapis.com/books/v1/volumes?q={query}&key=YOUR_API_KEY')
return response.json()
```

- **Integração com Sistema de Pagamentos**: Configuração para cobrança de taxas de atraso.
  - **Objetivo:** Permitir integração com sistemas de pagamentos para a cobrança de taxas de atraso.
  - **Implementação:**
    - Utilizar serviços de pagamento, como Stripe ou PayPal, para processar pagamentos.
    - Configurar endpoints para gerir transações e cobranças.

Monitorização e Logs

Configuração de ferramentas para monitorização e análise de logs:

- **Integração com Prometheus**: Monitorização do desempenho da aplicação.
  - **Objetivo:** Monitorizar o desempenho da aplicação.
  - **Implementação:**
    - **Instalar e configurar Prometheus** no servidor.
    - **Expor métricas** a partir da aplicação para serem recolhidas pelo Prometheus.
    - Exemplo de configuração no app.py:

```
python
from prometheus_client import start_http_server, Summary

REQUEST_TIME = Summary('request_processing_seconds', 'Time spent processing request')

@REQUEST_TIME.time()
def process_request():
  pass

if __name__ == '__main__':
  start_http_server(8000)
  app.run(debug=True)
```

- **Configuração do Elastic Stack**: Centralização e análise de logs.
    - **Objetivo:** Centralizar e analisar logs de atividades.
    - **Implementação:**
        - **Instalar e configurar Elastic Stack** (Elasticsearch, Logstash, Kibana) no servidor.
        - **Enviar logs** da aplicação para o Elasticsearch via Logstash.
        - **Visualizar e analisar logs** no Kibana.

Integração com Machine Learning

Descrição da integração com Machine Learning para recomendações:

- **Módulo de Recomendações**: Utilização de algoritmos de filtragem colaborativa.
    - **Objetivo:** Gerar recomendações de livros utilizando um modelo de Machine Learning.
    - **Implementação:**
        - Utilizar um algoritmo de filtragem colaborativa para gerar recomendações.
        - Exemplo de código para gerar recomendações:

```
python

from sklearn.neighbors import NearestNeighbors
import numpy as np

def recommend_books(user_id, user_history):
    # Exemplo de implementação simples
    model = NearestNeighbors(n_neighbors=5)
    model.fit(user_history)
    recommendations = model.kneighbors([user_id])
    return recommendations
```

- **Configuração do Endpoint de Recomendações**: Endpoint para retornar recomendações.
    - **Objetivo:** Retornar recomendações baseadas no histórico do utilizador.
    - **Implementação:**
        - **Endpoint:** /api/recommendations/user/:userId
        - Exemplo de configuração no Flask:

```
python

@app.route('/api/recommendations/user/<user_id>', methods=['GET'])
def get_recommendations(user_id):
    recommendations = recommend_books(user_id, user_history)
    return jsonify(recommendations)
```

Guia de Utilização

Instruções gerais para utilização e configuração das integrações:

- **Acesso e Configuração de Integrações**: Passos para configurar autenticação, sincronização de livros e pagamentos.
    - **Autenticação OAuth 2.0:**
        - **Registo e Credenciais:** Aceder ao Google Cloud Console para configurar as credenciais OAuth 2.0.
        - **Login:** Utilizar o botão "Login com Google" na interface web.
    - **API do Google Books:**
        - **Sincronização de Livros:** Utilizar a função fetch_books para buscar dados de livros.
    - **Sistema de Pagamentos:**
    - **Configuração de Pagamentos:** Integrar serviços como Stripe ou PayPal para processar cobranças de taxas de atraso
- **Monitorização e Logs**: Configuração de Prometheus e Elastic Stack.
    - **Prometheus:**
        - **Métricas de Desempenho:** Iniciar o servidor Prometheus e expor métricas a partir da aplicação.
    - **Elastic Stack:**
        - **Centralização de Logs:** Configurar Elasticsearch, Logstash e Kibana para centralizar e analisar logs.

- **Integração com Machine Learning**: Configuração do módulo de recomendações e endpoint.
    - **Recomendações de Livros:**
        - **Algoritmo: Utilizar a filtragem colaborativa para gerar recomendações.**
        - **Endpoint de Recomendações: Configurar o endpoint /api/recommendations/user/:userId para retornar recomendações baseadas no histórico do utilizador.**

Conclusão

Este guia de instalação e execução fornece todas as informações necessárias para configurar, implementar e validar a aplicação OnLib. Desde a preparação do ambiente com as ferramentas essenciais, passando pela configuração das bases de dados e do backend em Python com Flask, até à estruturação dos microserviços e integração com o RabbitMQ, cada passo é detalhado para garantir uma implementação bem-sucedida.

O frontend desenvolvido em React oferece uma interface moderna e intuitiva, enquanto a interface administrativa com Tkinter permite uma gestão eficiente da biblioteca. A integração com serviços externos, como a autenticação OAuth 2.0 e a API do Google Books, bem como a monitorização com Prometheus e Elastic Stack, asseguram que a aplicação é robusta e escalável.

A utilização de Machine Learning para recomendações personalizadas demonstra o compromisso com a inovação e a melhoria contínua da experiência do utilizador. Este guia não só facilita a instalação e execução da aplicação, mas também serve como um recurso abrangente para a sua manutenção e expansão futura.